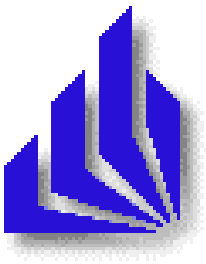

Universität-Gesamthochschule Paderborn

Diplomarbeit

Konzeption und Entwicklung eines Organisationsinterfaces für ein Ad-hoc-Workflow-Modellierungswerkzeug

Prof. Dr. L. Nastansky

Sommersemester 2001

Betreuer : Dipl. Inform. Carsten Huth

Vorgelegt von :

Marc Schwarzkopf
Wirtschaftsinformatik HII
Matr.-Nr. : 3595152
Bäckerstr. 20
32423 Minden / Westf.

und

Sebastian Scholz
Wirtschaftsinformatik HII
Matr.-Nr.: 3593986
Kasinoweg 25
34369 Hofgeismar

Inhaltsverzeichnis:

1 Einleitung ... 1

1.1 Szenario ... 1

1.2 Zielsetzung ... 2

1.3 Vorgehensweise ... 4

2 Grundlagen ... 6

2.1 Die Begriffe CSCW und Groupware ... 6

2.2 Workflowterminologie ... 9

2.2.1 Transaktionsorientiertes Workflow-Management ... 10

2.2.2 Ad-hoc-Workflow-Management ... 11

2.3 Organisation ... 12

2.3.1 Organisationsbegriff ... 12

2.3.1.1 Der funktionale Organisationsbegriff ... 13

2.3.1.2 Der instrumentale Organisationsbegriff ... 13

2.3.1.3 Der institutionale Organisationsbegriff ... 14

2.3.2 Aufbauorganisation ... 14

2.3.2.1 Stellen ... 15

2.3.2.2 Abteilungsstruktur ... 16

2.3.2.3 Arbeitsgruppenstruktur ... 20

2.3.3 Ablauforganisation ... 21

2.4 Das GroupProcess-Projekt ... 22

2.4.1 Konzepte und Ziele ... 22

3 Umgebung und Ressourcen ... 25

3.1 Groupwareumgebung Lotus Notes ... 25

3.2 Workflow-Management-System PAVONE Enterprise Office ... 27

3.2.1 Enterprise Office Organisation Database ... 28

3.2.1.1 Personen ... 29

3.2.1.2 Sachen ... 29

3.2.1.3 Abteilungen ... 29

3.2.1.4 Arbeitsgruppen ... 30

3.2.1.5 Rollen ... 31
3.2.2 Enterprise Office Workflows ... 31
3.2.2.1 Vordefinierte Workflows ... 32
3.2.2.2 Ad-hoc-Workflows ... 33
3.2.3 Umgebung und Ressourcen von GroupProcess ... 34
3.3 Java ... 37
3.3.1 Plattformunabhängigkeit ... 38
3.3.2 Sicherheitskonzepte ... 40
3.3.2.1 Sandbox ... 40
3.3.2.2 Signierte Applets ... 41
3.3.2.3 Protection Domains ... 41
3.3.3 Objektorientiertheit ... 42
3.4 Techniken der Benutzungsschnittstellen- und Dialoggestaltung ... 43
3.4.1 Benutzungsschnittstellen ... 44
3.4.2 Dialoggestaltung und -steuerung ... 46
3.5 Verschiedene Kommunikationsmöglichkeiten zwischen Java- und Nicht-Java-Applikationen ... 49
3.5.1 Kommunikation über Datenbanken ... 50
3.5.1.1 Kommunikation über relationale Datenbanken ... 50
3.5.1.2 Kommunikation über objektorientierte Datenbanken ... 53
3.5.2 Kommunikation über dateibasierte Schnittstellen ... 54
3.5.3 Kommunikation über spezielle Verbindungstechnologien ... 54
3.5.3.1 Kommunikation über CORBA ... 55
3.5.3.2 Kommunikation über COM ... 57
3.5.3.3 Vergleich von CORBA und COM ... 59
3.5.3.4 Extensible Markup Language (XML) ... 59
3.5.4 Spezielle Kommunikationsmöglichkeiten von Lotus Notes ... 61
3.6 Kommunikation zwischen Java-Applikationen ... 63
3.6.1 Kommunikation über RMI ... 63
3.6.2 Kommunikation über Interfaces ... 64

4 Anforderungen, Merkmale und Lösungsansatz eines OrganizationchartViewers und eines WorkgroupViewers für ein Ad-hoc-Workflow-Modellierungswerkzeug 67

4.1 Kontinuum der Nutzertypen von Organisationsdesignwerkzeugen 68

4.2 Benutzerinterface für einen OrganizationchartViewer und einen WorkgroupViewer 70

4.3 Kommunikation des OrganizationchartViewers / WorkgroupViewers mit dem GroupProcess-Projekt und Lotus Notes 72

4.3.1 Anbindung des Organizationchart- / WorkgroupViewers an das Ad-hoc-Workflow-Managementwerkzeug 72

4.3.2 Anbindung des GroupProcess-Projektes an Lotus Notes-Datenbanken . 73

5 Eine Java-basierte Lösung für einen Organizationchart- und einen WorkgroupViewer und die Anbindung an Lotus Notes 75

5.1 Prototypen des OrganizationchartViewers und des WorkgroupViewers 75

5.1.1 Dialogmodellierung 75

5.1.2 Optische Gestaltung 77

5.1.2.1 Die optische Gestaltung des OrganizationchartViewers 78

5.1.2.2 Die optische Gestaltung des WorkgroupViewers 79

5.2 Architektur des OrganizationchartViewers, des WorkgroupViewers und der Kommunikation 81

5.2.1 Die Architektur des OrganizationchartViewers 82

5.2.2 Die Architektur des WorkgroupViewers 83

5.2.3 Der Aufbau der verwendeten Kommunikationslösung 85

5.2.3.1 Kommunikation mit Lotus Notes 85

5.2.3.2 Kommunikation mit dem Ad-hoc-WorkflowModeler 85

5.3 Flexibilität und Einstellungsmöglichkeiten der Werkzeuge 86

5.3.1 Optische Anpassung der Komponenten 86

5.3.2 Flexibilität der Werkzeuge 88

6 Ausblick 89

7 Zusammenfassung 91

8 Literaturverzeichnis 93

Anhang 101

A.1 Anwendungsbeispiele des GroupProcess-Projektes 102

A.2 Anwendungsbeispiele der erstellten Werkzeuge 103

A.3 Systemvorraussetzungen und verwendete Hilfsmittel 104

A.3.1 Mindestanforderungen 104

A.3.2 Verwendete Hilfsmittel 105

A.4 Quelldateien im Überblick 106

A.4.1 Quelldateien des OrganizationchartViewers 106

A.4.2 Quelldateien des WorkgroupViewers 107

A.4.3 Quelldateien des Kommunikationspaketes readNotesData 107

A.4.4 Codebeispiel 108

Aufteilung der Kapitel auf die Autoren

Die vorliegende Arbeit wurde von Marc Schwarzkopf und Sebastian Scholz verfasst. Die folgende Tabelle zeigt die Verteilung der einzelnen Kapitel auf die jeweiligen Autoren. Die Angaben schließen das Anfangs- bzw. Endkapitel mit ein.

Kapitel	**Autor**
1	Sebastian Scholz
2	
2.1-2.2	Sebastian Scholz
2.3-2.4	Marc Schwarzkopf
3	
3.1-3.4	Marc Schwarzkopf
3.5-3.6	Sebastian Scholz
4	
4.1-4.3	Sebastian Scholz
5	
5.1	Marc Schwarzkopf
5.2-5.3	Sebastian Scholz
6	Marc Schwarzkopf
7	Marc Schwarzkopf
8 u. Anhang	Gemeinsam verfasst

Tabelle 1: Verteilung der Kapitel auf die Autoren

Abkürzungsverzeichnis:

API	Application Programming Interface
ASP	Application Service Providing
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CSCW	Computer Supported Cooperative Work
DCOM	Distributed Component Object Model
DIN	Deutsches Institut für Normung
HCI	Human Computer Interaction
HTML	Hypertext Markup Language
IDL	Interface Definition Language
IIOP	Internet Inter ORB Protocol
IuK-Technik	Informations- und Kommunikationstechnik
JDBC	Java Database Connect
LAN	Local Area Network
LRPC	Lightweight Remote Procedure Call
MIDL	Microsoft Interface Definition Language
NC	Network Computer
ODBC	Open Database Connect
OLE	Object Linking and Embedding
OMG	Object Management Group
OOP	Objekt Orientierte Programmierung
ORB	Object Request Broker
PC	Personal Computer
RMI	Remote Method Invocation

RPC	Remote Procedure Call
SGML	Standard Generalized Markup Language
SQL	Structured Query Language
UI	User Interface
VDI	Verein deutscher Ingenieure
VM	Virtual Machine
W3C	World Wide Web Consortium
WAN	Wide Area Network
WFMS	Workflow-Management-Systeme
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language

Verzeichnis der Abbildungen und Tabellen:

Abbildungen:

Abbildung 1: Klassifikationsschema nach Unterstützungsfunktionen 8
Abbildung 2: Das Referenzmodell der Workflow Management Coalition (WFMC) ... 10
Abbildung 3: Gängige Gruppierungsmöglichkeiten von Abteilungen 17
Abbildung 4: Das Einliniensystem 19
Abbildung 5: Das Mehrliniensystem 19
Abbildung 6: Das Matrixsystem 20
Abbildung 7: Integration von GroupProcess in Workflow-Management-Systeme..... 23
Abbildung 8: Lotus Notes-Verbunddokument 26
Abbildung 9: Zusammenhang der PAVONE Datenbanken 28
Abbildung 10: Die Darstellung der Abteilungen in PAVONE..... 30
Abbildung 11: Die Darstellung der Arbeitsgruppen in PAVONE 31
Abbildung 12: Der PAVONE ProcessModeler 32
Abbildung 13: Details eines Ad-hoc-Workflows in PAVONE Enterprise Office 33
Abbildung 14: Übersicht über den Workflow bei PAVONE Enterprise Office..... 34
Abbildung 15: GroupProcess - der Ad-hoc-WorkflowModeler 36
Abbildung 16: Die Virtual Machine von Java 39
Abbildung 17: Ein Objekt..... 42
Abbildung 18: Benutzungsoberfläche und Benutzungsschnittstelle in Kombination mit Anwendungssystemen 45
Abbildung 19: Kommunikation zwischen Applikationen und relationalen Datenbanksystemen 51
Abbildung 20: Kommunikation mit JDBC 52
Abbildung 21: Die Kommunikation über den ORB 56
Abbildung 22: Kommunikation zwischen zwei ORBs mit IIOP..... 57
Abbildung 23: COM-Interfaces 58
Abbildung 24: Ein XML Beispiel..... 61
Abbildung 25: Kommunikation über RMI 64
Abbildung 26: Dialognetz der Werkzeuge nach Janssen..... 76
Abbildung 27: Der 3D Effekt bei den Organisationseinheiten..... 77

Abbildung 28: Die Hilfeleiste 78
Abbildung 29: Der OrganizationchartViewer 78
Abbildung 30: Reduzieren des Hierarchiebaumes 79
Abbildung 31: Verschiedene Anordnungen beim WorkgroupViewer 80
Abbildung 32: Die Darstellung der Arbeitsgruppen 81
Abbildung 33: Die Packagestruktur des GroupProcess-Projektes 81
Abbildung 34: Klassendiagramm des OrganizationchartViewers 82
Abbildung 35: Die Klassenstruktur des WorkgroupViewers 84
Abbildung 36: Verschiedene Darstellungsmöglichkeiten des OrganizationchartViewers 87
Abbildung 37: Der Ad-hoc-WorkflowModeler in Kombination mit dem OrganizationchartViewer eingebettet in ein Lotus Notes-Dokument 102
Abbildung 38: Der Ad-hoc-WorkflowModeler in Kombination mit dem WorkgroupViewer eingebettet in ein Lotus Notes-Dokument 103
Abbildung 39: Der WorkgroupViewer 103
Abbildung 40: Der OrganizationchartViewer in verschiedenen Darstellungen 104
Abbildung 41: Codebeispiel der mouseDown-Routine im OrganizationchartViewer . 108

Tabellen:

Tabelle 1: Verteilung der Kapitel auf die Autoren VI
Tabelle 2: Varianten von Workflow-Management-Systemen 12
Tabelle 3: Klassenstruktur des OrganizationchartViewers 106
Tabelle 4: Klassenstruktur des WorkgroupViewers 107
Tabelle 5: Klassenstruktur des Kommunikationspackages 107

1 Einleitung

1.1 Szenario

Die veränderten Wettbewerbsbedingungen verlangen von den Unternehmen in Zukunft mehr Flexibilität und Innovationsfähigkeit statt Produktivitätssteigerung durch starre Arbeitsteilung, um am Markt weiterhin bestehen zu können. Notwendig ist eine Abflachung oder sogar eine Auflösung hierarchischer Strukturen. Klassische Abteilungen und Hierarchieebenen verlieren ihre Bedeutung, streng festgelegte Kommunikationsstrukturen werden ersetzt durch den direkten Weg einer nicht im einzelnen kanalisierten Gruppenkommunikation. Die Zusammenführung von dispositiver und objektbezogener Arbeit sowie die Zusammenführung von Dienstleistung und Sachleistung zu geschlossenen Wertschöpfungsketten hat aber noch eine weitere Konsequenz, welche die Grenzen der Unternehmung auch in räumlicher Hinsicht in Frage stellt: Je stärker das Konzept der autonomen Unternehmenseinheiten durch Informations- und Kommunikationstechniken unterstützt werden kann, desto stärker tritt auch die Standortfrage in den Vordergrund. Können mit einer Standortverlagerung ökonomische Vorteile erzielt werden, z.B. durch größere Marktnähe, durch die Nutzung von Kostenvorteilen, durch Erhöhung der Lebensqualität für die Mitarbeiter oder durch Versorgungsvorteile, dann folgt der organisatorischen Dezentralisierung auch die räumliche Dezentralisierung, d.h. die Standortverlagerung von Organisationseinheiten. Diese erstreckt sich auf die Standorte von ganzen Unternehmen, von modularen Organisationseinheiten, Gruppen oder einzelnen Arbeitsplätzen. Es ist daher nicht verwunderlich, dass im Zuge der Modularisierung und der Neustrukturierung der Unternehmensorganisation neue Formen der Arbeit und der Arbeitsteilung intensiv diskutiert werden.[1]

Auflösung der Wertschöpfungskette, Modularisierung, Netzwerke, elektronische Märkte und Standortunabhängigkeit sind die Schlagworte, die den Weg zu neuen Unternehmungsformen und zur Virtualisierung kennzeichnen. Dieser Weg ist freilich

[1] Vgl. Picot / Reichwald / Wigand 1998, S.10

ein offener Weg, auf dem Unternehmer - in diesem Fall organisatorische Innovatoren - mit Hilfe der neuen Möglichkeiten neue Koordinationsstrukturen ausprobieren und durchsetzen. Die von technischen Systemen zunehmend angebotene Möglichkeit, artikulierbare Informationen und explizites Wissen an jedem Ort zu jeder Zeit abrufen und austauschen zu können, hebt viele (nicht alle) Grenzen traditioneller Unternehmens- und Marktgestaltung auf.

Damit eröffnet die Nutzung der IuK-Technik neue Handlungsspielräume für die Gestaltung und den Betrieb von Wertschöpfungsprozessen, die den Weg zu virtuellen Unternehmen, die grenzenlos erscheinen, öffnen.[2]

Durch den Einsatz von Groupware haben Unternehmungen, insbesondere in der letzten halben Dekade, die Möglichkeit geschaffen, ihre Aufbauorganisation an die aktuellen Anforderungen anzupassen. Durch die Tendenz, zunehmend Aufgaben in Projekten durch den Einsatz von Arbeitsgruppen zu bearbeiten, hat die Bedeutung von teamunterstützender Informations- und Kommunikationstechnik stark zugenommen.

Mit der Entwicklung von GroupProcess unterstützt der Lehrstuhl Wirtschaftsinformatik 2 der Universität Paderborn diese Tendenzen und stellt Möglichkeiten zur Verfügung, dynamische und räumlich verteilte Strukturen in Unternehmungen zu modellieren. Da in Organisationen Prozesse auftreten können, die sich keiner festen Struktur zuordnen lassen, bietet das GroupProcess-Projekt Grundlagen zur situativen Gestaltung solcher Abläufe. Zur Gestaltung dieser Abläufe muss die Möglichkeit bestehen, zum einen auf die Organisationsstruktur einer Unternehmung und zum anderen auf einzelne Mitglieder von Organisationseinheiten zugreifen zu können.

1.2 Zielsetzung

Das Ziel der Arbeit ist, basierend auf einem Prototyp im Bereich der Modellierung von Ad-Hoc-Workflows die Integration von Informationen über Aufbauorganisation und Arbeitsgruppen in diesem Prototyp zu prüfen und zu realisieren.

---

[2] Vgl. Becker / Kloock / Schmidt / Wäscher 1998, S. 34

Es ergeben sich folgende drei Teilbereiche, die zur Erfüllung des Ziels bewältigt werden müssen:

Ein Teil beschäftigt sich mit der visuellen Darstellung der Abteilungs- und Arbeitsgruppenstruktur. In diesem Bereich können Mitglieder sowie ganze Abteilungen bzw. Gruppen ausgewählt werden und diese Auswahl dem bestehenden Prototyp zur Weiterverarbeitung übergeben werden.

Die Kommunikation zwischen den Visualisierungs- und Auswahlkomponenten und dem bereits existierenden Modellierungsprototypen wird in einem weiteren Teil behandelt. Hierbei werden unter anderem die Kommunikationsmöglichkeiten zwischen zwei oder mehreren Java-Komponenten untersucht.

Um die Aufbauorganisations- bzw. Arbeitsgruppenstruktur und deren Mitglieder in den Benutzeroberflächen darstellen zu können, müssen die entsprechenden Daten aus dem aktuellen Datenbestand vorhandener nicht Java-basierter Applikationen entnommen werden können. Ein letzter Teil beschäftigt sich daher mit der Kommunikation zwischen Java und anderen Applikationen und stellt die derzeitigen Möglichkeiten auf diesem Sektor dar.

Durch den möglichen Einsatz der Werkzeuge in den unterschiedlichsten Bereichen einer Unternehmung muss besonders bei der Modellierung der Benutzungsschnittstellen und der Dialoggestaltung darauf geachtet werden, dass unterschiedliche Nutzertypen für die Anwendung in Frage kommen. Daraus ergeben sich hohe Anforderungen insbesondere an die intuitive Bedienbarkeit. Aspekte aus dem Bereich der Oberflächengestaltung müssen in diesem Kontext mit den geforderten Funktionalitäten in Einklang gebracht werden.

Der Einsatz dieser Modellierungskomponenten auf der Basis verschiedener Betriebssysteme erfordert deren Portierbarkeit. In diesem Zusammenhang werden die Vorteile der Java-Technologie herausgearbeitet und dargestellt.

1.3 Vorgehensweise

Im direkt anschließenden zweiten Kapitel werden die Grundlagen und Vorraussetzungen erläutert. Die Einführung in die Thematik der Bereiche Groupware, Organisation und GroupProcess bildet den grundlegenden Rahmen für diese Arbeit und zeigt einen Überblick über die aktuellen Entwicklungen in den oben genannten Bereichen.

Die im dritten Kapitel betrachteten Umgebungen spielen eine große Rolle bei der Umsetzung des praktischen Teils dieser Arbeit, welcher im fünften Kapitel näher dargestellt wird. Daher deckt dieses Kapitel die Themen Lotus Notes, PAVONE Enterprise Office sowie GroupProcess, Java- und Kommunikationstechnologien im Rahmen der Erläuterung von Umgebungen und Ressourcen ab. Den Anforderungen an Benutzungsschnittstellen und an die Dialogmodellierung kommt im Rahmen dieser Arbeit eine hohe Bedeutung zu, um zum einen eine hohe Flexibilität der Visualisierungskomponenten zu erreichen und zum anderen deren Bedienbarkeit im Hinblick auf die Art der zu erwartenden Nutzer nicht einzuschränken.

Der Punkt Kommunikationstechnologien umfasst die Kommunikationsstrategien zwischen Java-basierten Applikationen und die daraus resultierenden Konsequenzen für die Datenstrukturierung innerhalb der Java-Klassen. Ein weiterer Schwerpunkt dieses Kapitels ist der Datenaustausch zwischen Java-basierten und Nicht-Java-basierten Anwendungen. Die Begriffe Remote Method Invocation (RMI) und Common Object Request Broker Architecture (CORBA) werden in diesem Bereich betrachtet werden. Traditionell angewandte Methoden der Datenhaltung und des Datenaustausches werden auch im Rahmen dieses Abschnittes erwähnt, wobei relationale Datenbanken und temporäre Dateien eine Rolle spielen werden. Zum Schluss dieses Kapitels wird auf die speziell von Lotus Notes bereitgestellten Java-basierten Kommunikationsmöglichkeiten eingegangen, die auch bei der späteren Realisation des Datentransfers zwischen Lotus Notes und den Java-basierten Werkzeugen zur Arbeitsgruppen- und Abteilungsverwaltung Anwendung finden werden. In diesem Teil des Kapitels werden auch andere von Lotus Notes bereitgestellte Methoden der Kommunikation, die nicht auf der Java-Technologie basieren, erörtert.

Im daran anschließenden vierten Kapitel wird analysiert, welche Arten von Nutzern die Workflow-Modellierungswerkzeuge in Unternehmen überwiegend anwenden werden. Danach werden die einzelnen, im dritten Kapitel erwähnten Möglichkeiten zur Kommunikation zwischen den Komponenten gegeneinander abgewogen, um abschließend Lösungsansätze für die Modellierung der Benutzungsschnittstellen und für die Kommunikation zwischen den einzelnen Komponenten zu finden.

Nachdem im fünften Kapitel die entwickelten Prototypen vorgestellt wurden, wird auf die Architektur des Organizationchart- und des WorkgroupViewers und den daraus resultierenden Veränderungen in der Basisapplikation Ad-hoc-WorkflowModeler eingegangen. Die Benutzeroberflächen der beiden Werkzeuge werden vorgestellt sowie die Interoperation mit dem Benutzer. Basierend auf den im dritten Kapitel vorgestellten Lösungsmöglichkeiten zur Kommunikation zwischen Java-Applikationen und Lotus Notes wird die in der Realisation angewandte Strategie aufgezeigt. Darüber hinaus werden optionale Anpassungs- und Einstellmöglichkeiten der in dieser Arbeit entwickelten Module beschrieben.

Ein Ausblick auf die in Zukunft zu erwartenden Entwicklungen in dieser Thematik bietet das siebte Kapitel.

Mit einer Zusammenfassung findet diese Arbeit ihren Abschluss.

2 Grundlagen

Ein Überblick über die Begriffe Computer Supported Cooperative Work (CSCW) und Groupware, die grundlegend wichtig für diese Arbeit sind, soll in dem nun folgenden Kapitel gegeben werden. Der Begriff Workflow sowie das von Dipl. Inform. Carsten Huth und Professor Dr. Ludwig Nastansky entwickelte GroupProcess-Projekt fließen in diese Arbeit ein. Daher werden diese Punkte in diesem Kapitel grundlegend erörtert. Das dritte Kapitel beschäftigt sich dann eingehender mit der Umgebung und den verwendeten Ressourcen.

2.1 Die Begriffe CSCW und Groupware

Der Begriff Computer Supported Cooperative Work (CSCW) wird in der Regel mit „Computergestützte Gruppenarbeit“ übersetzt und bezeichnet Applikationen, die in ihrer Gesamtheit als „Groupware“ zusammengefaßt werden. Die Nutzung dieser Systeme wird als „Workgroup Computing“ tituliert.[3]

Unter dem Begriff der computergestützten Gruppenarbeit werden Bemühungen in Forschung und Praxis verstanden, Informations- und Kommunikationstechnologie zur Unterstützung von arbeitsteiligen Prozessen einzusetzen. Objekte dieser Arbeit sind grundlegend Informationen, die sich in Dokumenten, Entscheidungen und Software finden lassen.[4]

Im Allgemeinen wird CSCW als ein junges Forschungsgebiet innerhalb der Informatik angesehen, dessen gesamte Tragweite derzeit noch nicht abzusehen ist.

> „CSCW is not particularly well defined, a consequence of its youth and its multi-disciplinary nature.“[5]

Die unter dem Fachausdruck Groupware zusammengefaßten Arbeitsmittel sind von sehr unterschiedlicher Gestalt und Komplexität, so dass eine leistungs- und

[3] Vgl. Knittel 1995, S. 144

[4] Vgl. Gabler 1998, S. 136

[5] Greenberg, 1991,S. 1

funktionsbezogene Einordnung dieses Begriffes nicht möglich ist. Was Groupware ausmacht, ist die spezifische Sicht auf die Erledigung der von ihr unterstützten Arbeit.

„The focus is on designing for the structure of the group activity“[6]

Demnach werden unter Groupware EDV-Werkzeuge zusammengefasst, die nur im Gruppenzusammenhang sinnvoll eingesetzt werden können, jedoch von einem Individuum alleine nicht oder nur begrenzt.

E-Mail als ein Beispiel für klassische Systeme zur Kommunikationsunterstützung ist zur Unterstützung von 1:1-Kommunikation und 1:n-Kommunikation geeignet. Im Rahmen der m:n Kommunikation kann ein E-Mail-System aufgrund der Komplexität des Nachrichtenaustausches allerdings nicht mehr effizient eingesetzt werden. Um Abhilfe in diesem Bereich zu schaffen, verwenden Groupwaresysteme wie z.B. Lotus Notes das Pull-Modell, um Informationen miteinander zu teilen, gemeinsam zu pflegen, zu strukturieren, gezielt in Kontexte einzubetten und weiterzuentwickeln.[7]

Im Szenario Groupware werden vier Systemklassen unterschieden :

Unter der Systemklasse *Kommunikation* werden Systeme verstanden, die zum einen Kanäle zum Austausch von Audio- und Videosignalen anbieten und zum anderen elektronische Post unterstützen. Die technische Verbesserung dieser Systemklasse liegt nicht mehr im Schwerpunkt heutiger CSCW-Forschung.[8]

Die Systemklasse *gemeinsame Informationsräume* stellt den Nutzern von Groupware-Systemen Applikationen zur Verfügung, in denen Daten verwaltet und verteilt werden. Diesen Applikationen kommt eine zentrale Bedeutung im Bereich Groupware zu, da Gruppenarbeit in vielen Fällen die gemeinsame Nutzung derselben Daten durch die Gruppenmitglieder erfordert.[9]

In der Systemklasse *Workflow Management* werden Systeme zusammengefasst, die zur Unterstützung betrieblicher Abläufe einsetzbar sind. Die Unterstützung von Workflow-

[6] Winograd 1989, S. 198

[7] Vgl. Nastansky et al. 2000, S.241f

[8] Vgl. Teufel 1995, S. 129

[9] Vgl. Teufel 1995, S. 153

Management-Systemen (WFMS) bezieht sich hauptsächlich auf die Koordination räumlich oder zeitlich verteilter Arbeitsgruppen.[10]

Die letzte Systemklasse beschäftigt sich mit der Klasse *Workgroup Computing*. Diese Systeme unterstützen die Kooperation von Personen, die in Gruppen arbeiten und Aufgaben zu erfüllen haben, welche durch einen mittleren bis geringen Strukturierungsgrad und eine niedrige Wiederholungsfrequenz gekennzeichnet sind.[11]

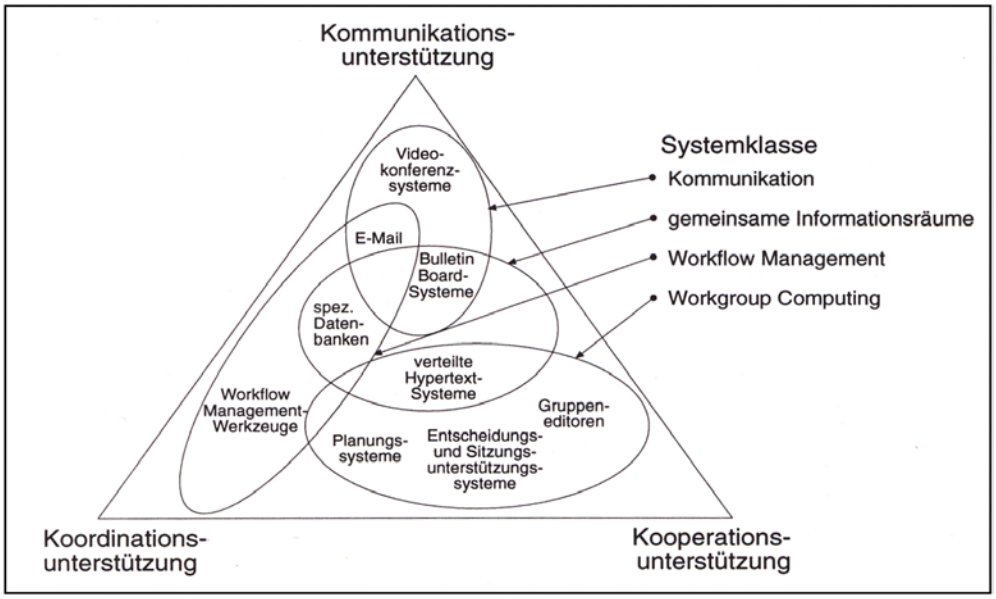

Abbildung 1: Klassifikationsschema nach Unterstützungsfunktionen[12]

Die oben gezeigte Abbildung zeigt die Einordnung der vier Systemklassen im Kontext der drei Unterstützungsrichtungen :

Wie alle Unterstützungsarten zielt auch die Kommunikationsunterstützung darauf ab, in Groupware-Systemen Informationen derart auszutauschen, dass das Handeln in Bezug auf die gegebenen Ziele optimal unterstützt wird.[13]

[10] Vgl. Teufel 1995, S. 181

[11] Vgl. Teufel 1995, S. 209

[12] Vgl. Teufel 1995, S. 127

[13] Vgl. Gabler 1998, S. 370

Koordinationsunterstützung bezeichnet die Fähigkeit von Groupware-Systemen, unterschiedliche organisatorische Einheiten so aufeinander abzustimmen, dass ein reibungsloser Ablauf entsteht.

Kooperation baut auf dem Kommunikationsbegriff auf und stellt den Austausch von Informationen mit einem gemeinsamen Ziel dar. Kooperation bedingt, dass mindestens zwei Personen in einem gemeinsamen, zielgerichteten Kooperationsprozess involviert sind.[14]

2.2 Workflowterminologie

Workflow-Management befasst sich damit, der richtigen Person zur richtigen Zeit die richtigen Informationen mit dem richtigen Werkzeug zur Verfügung zu stellen :

> „Ein Workflow verbindet die einzelnen Aufgaben eines Prozesses zu einem Ablauf und definiert, wer welche Aufgabe mit welchem Mittel und welchen Informationen durchführt. Eine Aufgabe kann selber in einzelne Arbeitsschritte aufgeteilt werden, welche manuell oder mit Hilfe von Anwendungen abgearbeitet werden.“[15]

Innerhalb eines Workflow-Management-Systems spielt die Workflow-Engine eine zentrale Rolle. Sie beschäftigt sich mit Steuerung und Kontrolle der einzelnen Aktivitäten, wie z.B. dem Weiterleiten von Informationen, dem Aufrufen von benötigten Applikationen und dem Sammeln von Statusinformationen des Workflows. Mit dieser Engine sind fünf Komponenten über standardisierte Schnittstellen verbunden. Die folgende Grafik zeigt diese Zusammenhänge :

[14] Vgl. Nastansky et al. 2000, S. 240ff

[15] Halter 1997, S. 174

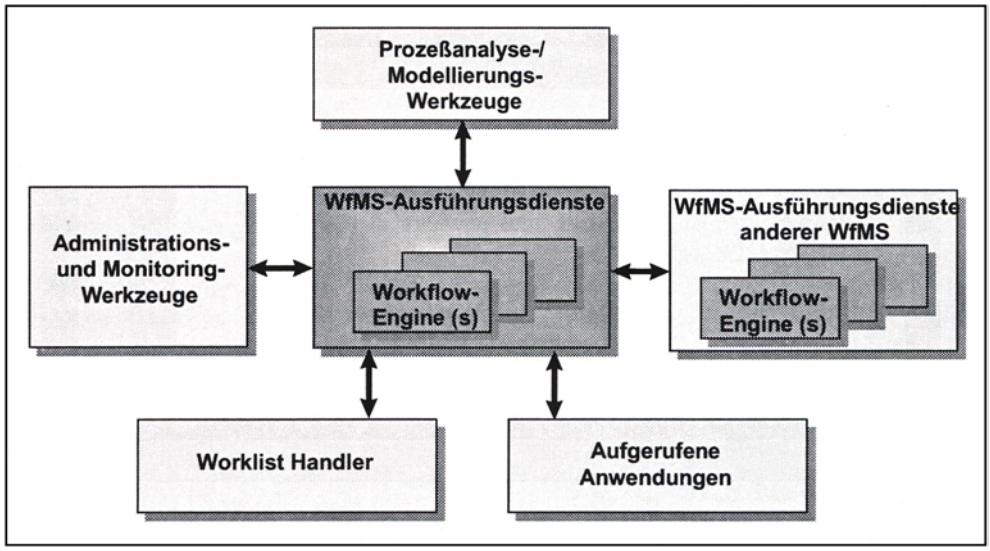

Abbildung 2: Das Referenzmodell der Workflow Management Coalition (WFMC) [16]

Workflow-Management-Systeme werden grundlegend in zwei Gruppen eingeteilt: zum einen in die transaktionsorientierten Workflow-Management-Systeme und zum anderen in die Ad-hoc-Workflow-Management-Systeme, die im Folgenden näher beschrieben werden.

2.2.1 Transaktionsorientiertes Workflow-Management

Im Kontext des transaktionsorientierten Workflow-Managements werden Prozesse beziehungsweise Abläufe betrachtet, die gut strukturiert sind und durch unterschiedliche Regeln und Prozeduren gut beschrieben und definiert werden können. Das bedeutet, dass sich die betrachteten Prozesse häufig wiederholen, ohne ihre Ablaufstruktur zu verändern. Aufgrund der guten Strukturierbarkeit dieser Prozesse werden deren Strukturen im Voraus in allen Details (Prozessabläufe, Vorgangs- und Statusprüfungen, Aufgabenzuordnungen und -prioritäten) in Workflow-Management-Systemen festgelegt. Die innerhalb der Systeme vorgegebene Struktur und die dazugehörigen Details können von den zugehörigen Bearbeitern kaum mehr beeinflusst werden. Zum Entwickeln solcher transaktionsorientierter Workflows stehen eine Reihe von Entwicklungs-

[16] Thielemann 1999, S. 33

werkzeugen zur Verfügung, wie zum Beispiel das Produkt „Enterprise Office“ von PAVONE, auf das im Punkt 3.2 näher eingegangen wird. Diese Entwicklungswerkzeuge sind in der Regel kompliziert in ihrer Handhabung und enthalten eine Vielzahl von Optionen und Optimierungsmöglichkeiten, so dass ungeschulte Mitarbeiter Schwierigkeiten mit ihrer Bedienung haben können.[17]

2.2.2 Ad-hoc-Workflow-Management

Im Ad-hoc-Workflow-Kontext werden Prozessflüsse behandelt, deren Struktur eher schlecht gestaltet ist oder deren Wiederholungshäufigkeit gering ist. Beim Auftreten dieser Abläufe ist es daher notwendig, in einem zugehörigen Workflow-Management-System eine Funktion bereitzustellen, mit der es möglich ist, solche spontan auftretenden Prozessflüsse zu entwerfen oder bereits ähnlich strukturierte Abläufe entsprechend anzupassen. Im Rahmen von Ad-hoc-Workflows kommt es häufig vor, dass sich bereits erzeugte Abläufe während der Abarbeitung noch in ihrer Struktur ändern können, weshalb eine solche Funktion einfach gestaltet und dezentral verfügbar sein sollte. So ist einer Mehrzahl der Bearbeiter aufgrund ihrer Kenntnisse die Möglichkeit gegeben, ein solches Werkzeug intuitiv zu bedienen. Wegen dieser Möglichkeiten sind solche Ad-hoc-Workflow-Managementwerkzeuge flexibler als transaktionsorientierte.[18]

Auch in diesem Bereich gibt es bereits eine Lösung in „Enterprise Office“ von PAVONE, auf die im Punkt 3.2.2.2 weiter eingegangen wird.

Einen grundlegenden Überblick über die Anforderungen an die beiden Arten des Workflow-Managements liefert die folgende Tabelle :

[17] Vgl. Haberstock 2000, S. 79

[18] Vgl. Nastansky et al. 2000, S. 244f

Merkmale	Transaktionsorientiertes Workflow Management	Ad-hoc Workflow Management
Regelgebundenheit	Hoch	Gering
Abläufe	Determiniert und im System definiert	Zusätzlich von Mitarbeitern beeinflußbar
Vorgangs- und Statusprüfung	Prüfung erfolgt im System nach Regeln	Zusätzlich von Mitarbeitern beeinflußbar
Verantwortlicheiten und Zuständigkeiten	Determiniert und im System definiert	Zusätzlich von Mitarbeitern beeinflußbar
Prioritäten und Termine	Determiniert und im System definiert	System unterstützt Mitarbeiter
Mitarbeiter	Anleitung und Führung durch das System	Nutzung des Systems als Werkzeug und Informationsressource

Tabelle 2: Varianten von Workflow-Management-Systemen[19]

2.3 Organisation

2.3.1 Organisationsbegriff

Eine Definition des Organisationsbegriffes im Allgemeinen ist aus der Literatur nicht zu entnehmen. Vielmehr existieren mehrere Interpretationen dieses Begriffes, deren Bedeutung dem entsprechenden Kontext zu entnehmen ist. Aus etymologischer Sichtweise betrachtet, bedeutet „Organisation“ das planmäßige Gestalten eines organischen Ganzen mit einer gefügehaften Ordnung. Dies gilt sowohl für den Naturbereich als auch für den Kulturbereich (Gesellschafts-, Sozial- und Humanbereich), der als Ergebnis der menschlichen Gestaltungstätigkeit betrachtet werden kann.

Löst man den Teilbereich Wirtschaft aus dem Kulturbereich heraus, so bleibt der allgemeine Organisationsbegriff auch hierfür gültig, jedoch mit dem Unterschied, dass zusätzlich noch die Bemühungen um Wirtschaftlichkeit mit einbezogen werden müssen.[20]

Für den allgemeinen Organisationsbegriff sind somit drei Merkmale wesentlich:

[19] Vgl. Nastansky et al. 2000, S. 245

[20] Vgl. Weidner 1977, S12

- Ordnung
- Zielgerichtetheit
- Wirtschaftlichkeit

Der Begriff der unternehmensbezogenen Organisation wird in der Literatur in drei Kategorien eingeteilt, die im Folgenden erläutert werden.

2.3.1.1 Der funktionale Organisationsbegriff

Unter dem funktionalen Organisationsbegriff wird der Prozess verstanden, der das Formulieren genereller Regeln zum Ziel hat, die das arbeitsteilige Vorgehen und damit das Zusammenwirken verschiedener Personen und Einheiten möglichst zielwirksam gestalten. In diesem Sinne wird Organisation als Mittel gesehen, das von der Betriebs- und Geschäftsleitung eingesetzt wird, um eine Organisationsstruktur zu schaffen oder zu reorganisieren.[21]

2.3.1.2 Der instrumentale Organisationsbegriff

Als Ergebnis der Organisation im funktionalen Sinne entsteht eine Organisation in instrumentaler Sichtweise. Somit versteht man die Organisation als Prozessergebnis. Sie ist die Gesamtheit aller Regelungen und Strukturen, die zur bestmöglichen Zielerreichung notwendig sind. Dazu gehört einerseits die sogenannte Aufbaustruktur[22], welche die Regelung von Aufgaben, Kompetenzen und Unterstellungsverhältnissen umfasst und andererseits die Ablaufstruktur, in der Regelungen bezüglich der Prozessgestaltung zusammengefasst werden. Die Fokussierung liegt demnach auf der Kernaussage : „Die Unternehmung *hat* eine Organisation“.[23]

---

[21] Vgl. Bühner 1989, S1ff

[22] Auf verschiedene Aufbauorganisationen wird im Kapitel 2.3.2 näher eingegangen.

[23] Vgl. Krüger S13, 1994

2.3.1.3 Der institutionale Organisationsbegriff

Beim institutionalen Organisationsbegriff wird die Organisation hingegen als soziales Gebilde verstanden, also als Oberbegriff für Institutionen wie Unternehmungen, öffentliche Dienstleistungsbetriebe, Verwaltungen, Spitäler, Schulen, Gefängnisse, Kirchen, politische Parteien, Verbände, Vereine, usw.[24] Die Interpretation von organisierten Sozialsystemen als Organisation wird also in diesem Sinne als „Die Organisation *ist* eine Unternehmung“ aufgefasst.

2.3.2 Aufbauorganisation

Die Aufbauorganisation, auch Strukturorganisation genannt, befasst sich mit der Gestaltung der Unternehmensstruktur. Ihre Aufgabenstellung beinhaltet daher folglich :

- Die Gestaltung der Systemelemente
- Die Gestaltung der hierarchischen Systembeziehungen

Die Gestaltung dieser Organisationsstruktur erfolgt in zwei Stufen, der Aufgabenanalyse und der Aufgabensynthese. Zweck der Aufgabenanalyse ist es, zu untersuchen, welche Tätigkeiten vorgenommen werden müssen, damit die Ziele der Organisation erreicht werden. Im Rahmen der Aufgabenanalyse wird von der Gesamtaufgabe ausgegangen und diese in mehrere Teilaufgaben gegliedert, wobei eine Baumstruktur der Aufgabenstruktur entsteht. Die Tiefe dieser Struktur ist dabei abhängig von:

[24] Vgl. Hill / Fehlbaum / Ulrich, 1994, S 17

- der aufbauorganisatorischen Aufgabenstellung
- der Komplexität der Aufgaben
- dem Grad der gewünschten Arbeitsteilung
- dem Sachmitteleinsatz
- der Häufigkeit des Aufgabenanfalls

Das Ergebnis dieser Analyse ist der Aufgabengliederungsplan. Er gliedert die Gesamtaufgabe in Haupt-, Teil- und Elementaraufgaben.

Durch die Aufgabensynthese werden die durch die Aufgabenanalyse erzeugten Elementaraufgaben zu Stellen zusammengefasst.[25]

2.3.2.1 Stellen

Stellen gelten als Grundelemente (Basiselemente) der Aufbauorganisation. Eine Stelle wird im Wesentlichen durch die im Folgenden aufgeführten Merkmale gekennzeichnet:

1. *Stellenaufgabe*: Durch die Zuordnung eines bestimmten Aufgabenbereichs wird der Stelle eine Aufgabe zur Ausführung übertragen. Eine Stelle grenzt somit aufgabenmäßig auf längere Sicht einen bestimmten Zuständigkeitsbereich ab.
2. *Stelleninhaber*: Jede Stelle muss durch mindestens eine Person als Stelleninhaber besetzt werden.
3. *Dauerhaftigkeit*: Stellen werden auf längere Sicht festgelegt. Deshalb werden sie üblicherweise nicht auf bestimmte Personen fixiert, sondern auf gedachte Stelleninhaber mit einer normalen Eignung und einem normalen Leistungsvermögen.
4. *Kompetenzen*: Zur ordnungsgemäßen Erfüllung der Stellenaufgabe wird der Stelleninhaber mit entsprechenden Kompetenzen, d.h. mit formalen Rechten und Befugnissen ausgestattet. Dazu gehören Durchführungs- und Leitungskompetenzen.

[25] Vgl. Steinbuch 1977, S 163 ff

5. *Verantwortung*: Ein Stelleninhaber ist verantwortlich für die in seinem Einflussbereich liegenden Tätigkeiten und Ereignisse. Dieser Bereich ergibt sich durch die Zuteilung von Aufgaben und Kompetenzen.[26]

Die Zusammenfassung von Stellen führt zu Stellenmehrheiten. Die Bildung von Stellenmehrheiten kann dabei entweder nach dem Hierarchieprinzip oder dem Gruppenprinzip erfolgen. Bei Verwendung des Hierarchieprinzips bilden die Stellenmehrheiten sogenannte Abteilungen und Abteilungssysteme. Kommt hingegen das Gruppenprinzip zur Anwendung, werden aus den zusammengesetzten Stellen Arbeitsgruppen und Arbeitsgruppensysteme erzeugt.[27]

2.3.2.2 Abteilungsstruktur

Im einfachsten Fall bestehen Abteilungen aus einer Instanz (z.B. Abteilungsleiter) und den ihr zugeordneten Ausführungsstellen (z.B. Sachbearbeiter). Abteilungen können wiederum aus Untereinheiten bestehen oder zu größeren Einheiten gebündelt werden. Des Weiteren ist es möglich, innerhalb von einzelnen Abteilungen Teams einzusetzen, wenn z.B. ein hoher Anteil innovativer Aufgaben zu erledigen ist (z.B. Forschung und Entwicklung). Außerdem besteht die Möglichkeit, den Abteilungen Stabsstellen oder Dienstleistungsstellen (z.B. Sekretariat) zuzuweisen.[28]

Bei der Bildung von Abteilungen sollten das Homogenitäts- und das Beherrschbarkeitsprinzip zur Anwendung kommen:

[26] Vgl. Schulte Zurhausen 1995, S 127ff

[27] Vgl. Bühner 1989, S 84f

[28] Vgl. Krüger 1994, S. 53f

- Das *Homogenitätsprinzip* besagt, dass in einer Abteilung solche Stellen zusammengefasst werden, deren Aufgaben eine hohe Zusammengehörigkeit aufweisen. Ziel ist es hierbei, die erforderliche abteilungsübergreifende Koordination zu minimieren. Die Abteilungen sollen deshalb ihren Aufgabenbereich so autonom wie möglich wahrnehmen können.
- Das *Beherrschbarkeitsprinzip* besagt, dass in einer Abteilung nur so viele Stellen zusammengefasst werden sollen, dass der Abteilungsleiter seinen Leitungsaufgaben nachkommen kann.

Zur Gruppierung von Stellen zu Abteilungen können wie bei der Stellenbildung verschiedene Kriterien genutzt werden. Die folgende Abbildung gibt einen Überblick über die gängigsten Gruppierungsmöglichkeiten.

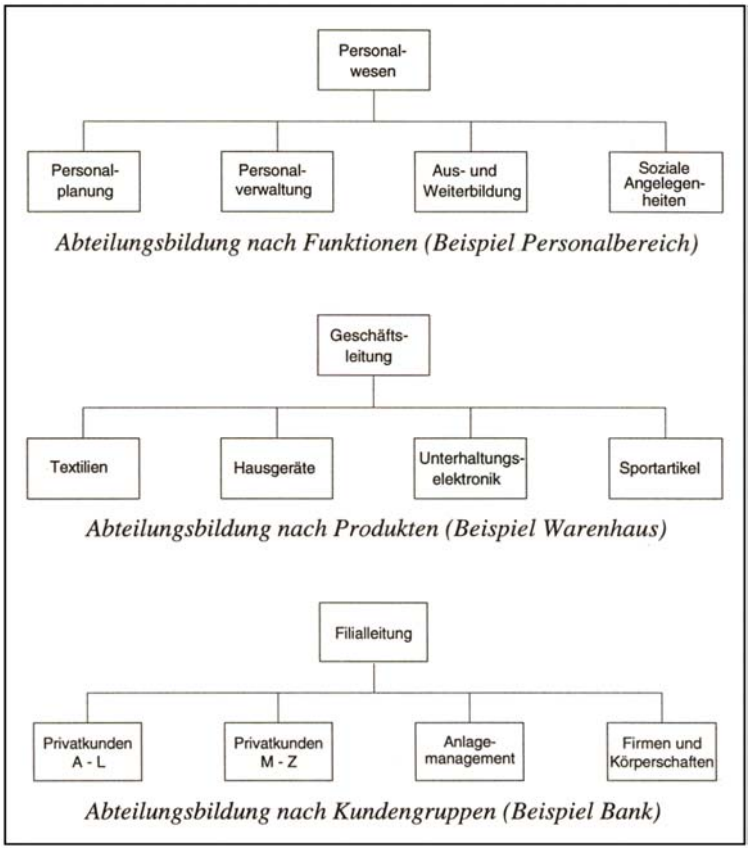

Abbildung 3: Gängige Gruppierungsmöglichkeiten von Abteilungen

Bei der *Abteilungsbildung nach Funktionen* werden nur Stellen zusammengefasst, die gleiche oder verwandte Funktionen ausüben. Daraus resultiert, dass innerhalb der Abteilung nahezu homogene Personalqualifikationen bestehen und ähnliche Methoden und Sachmittel Anwendung finden.

Findet eine *Abteilungsbildung nach Produkten* statt, so werden jene Stellen zusammengefasst, die sich mit denselben Produkten oder Dienstleistungen beschäftigen. Es ist nicht ausgeschlossen, dass in den so geschaffenen Abteilungen wiederum Unterabteilungen existieren, die nach einem anderen Kriterium zusammengeschlossen worden sind.

Wie oben gezeigt, gruppiert die *Abteilungsbildung nach Kundengruppen* nach Kundenmerkmalen, um deren speziellen Wünschen gerecht zu werden (z.B. Bankgewerbe). So kann z.B. eine Bank nach Privatkunden und nach Firmen und Körperschaften segmentiert werden.[29]

Die so zusammengefassten Organisationseinheiten benötigen Kommunikationsbeziehungen zueinander, um Informationen über die Entscheidungsinterdependenzen auszutauschen. Innerhalb der betreffenden Organisation bestehen grundsätzlich rangmäßige Über- und Unterordnungsverhältnisse (Hierarchie). Zur Ausformung dieser rangmäßigen Beziehungen zwischen den Organisationseinheiten stehen drei klassische Prinzipien zur Verfügung: das *Einliniensystem*, das *Mehrliniensystem* und das *Matrixsystem*.

[29] Vgl. Schulte-Zurhausen 1995, S.171ff

1. Das *Einliniensystem* basiert auf dem Prinzip, dass jede untergeordnete Organisationseinheit jeweils nur von einer übergeordneten Weisungen erhalten kann.

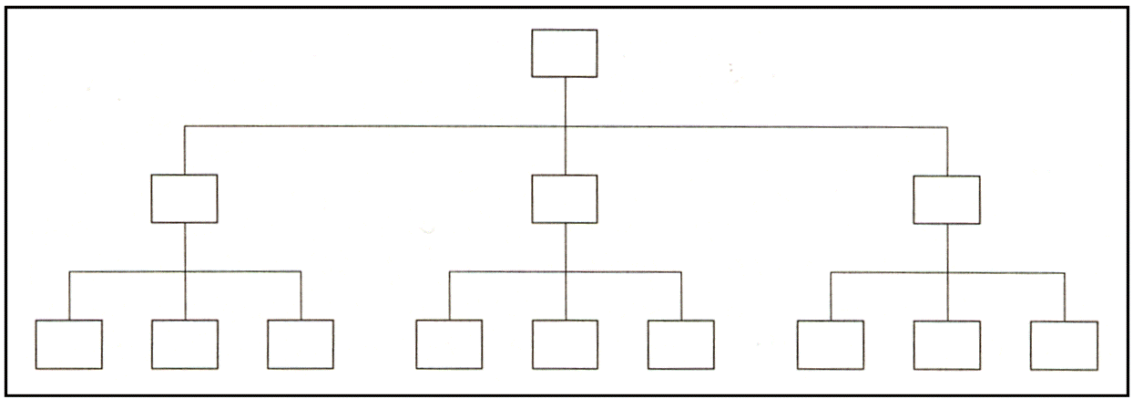

Abbildung 4: Das Einliniensystem

2. Im Gegensatz dazu begründet sich das *Mehrliniensystem* auf der Tatsache, dass eine untergeordnete Organisationseinheit von mindestens zwei anderen Einheiten Weisungen erhalten kann. Dieses Prinzip wird auch als Prinzip des kürzesten Weges bezeichnet, da sich die Mitarbeiter mit ihren Problemen direkt an die jeweiligen Spezialisten wenden können.

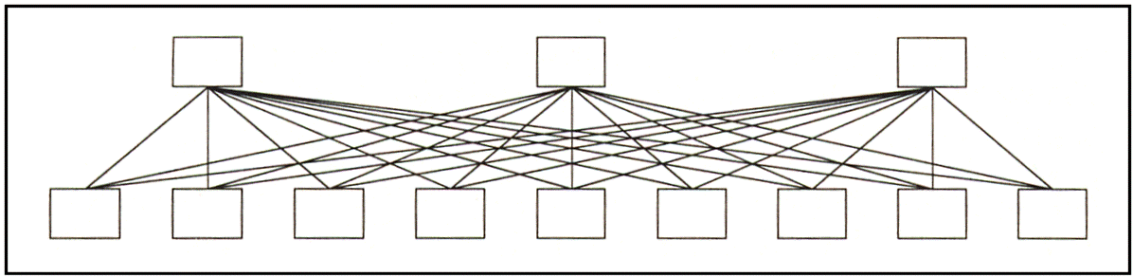

Abbildung 5: Das Mehrliniensystem

3. Beim *Matrixsystem* werden die Leitungsfunktionen auf zwei Matrixstellen aufgeteilt. In der Regel ist eine Leitungsdimension funktionsorientiert und die andere objektorientiert. Die Matrixschnittstellen sind in diesem System die untergeordneten Stellen und erhalten von zwei übergeordneten Stellen ihre Weisungen – das Matrixsystem wird deshalb auch als spezielles Mehrliniensystem bezeichnet.

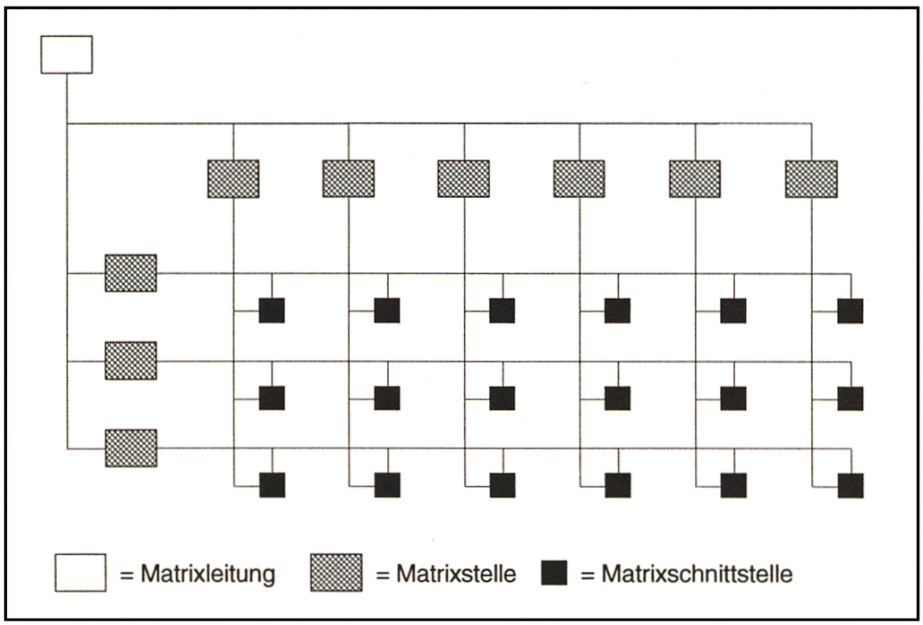

Abbildung 6: Das Matrixsystem

Durch die zunehmende Bedeutung von Gruppenarbeit gewinnt auch die Frage der organisatorischen Einbindung von Arbeitsgruppen in die hierarchische Organisationsstruktur zunehmend an Gewicht. Im Folgenden wird daher auf mögliche Strukturierungen innerhalb von Arbeitsgruppen eingegangen.

2.3.2.3 Arbeitsgruppenstruktur

Die beiden wesentlichen Modelle zur Integration von Arbeitsgruppen in die Unternehmensorganisation sind zum einen das *System sich überlappender Gruppen* und zum anderen das *Konzept vermaschter Teams*.

1. Das *System sich überlappender Gruppen* besteht aus einzelnen Gruppen, die jeweils aus Gruppenmitgliedern und einem Gruppenleiter bestehen. Die Gruppenleiter sind auf der nächsthöheren Ebene einfaches Mitglied einer weiteren Gruppe. Die Personen, die mehr als einer Gruppe angehören (Gruppenleiter), können in diesem System als Verbindungsglieder zwischen den Gruppen angesehen werden und lösen die traditionellen Vorgesetzten ab. Kommunikation zwischen Arbeitsgruppen kann somit intensiver und über Ebenen hinweg stattfinden.

2. Das *Konzept vermaschter Teams* stellt eine Kombination von Projektgruppe und Ausschuss dar. Die konventionelle Linienorganisation wird durch zusätzliche Arbeitsgruppen ergänzt, die hauptsächlich die Bearbeitung von komplexen Aufgaben mit innovativem und kreativem Lösungsgehalt vornehmen. Bei größeren Projekten und der damit verbunden Anzahl an Bearbeitern kann es notwendig sein, mehrere parallel arbeitende Teams zu bilden. Die Koordination dieser Teams erfolgt über zu bildende Planungsteams. Um zu erreichen, dass der Gesamtzusammenhang nicht verloren geht, erfolgt die Koordination der Planungsteams über deren Vermaschung.[30]

2.3.3 Ablauforganisation

„Gegenstand der Ablauforganisation ist der Ablauf des betrieblichen Geschehens und die Ausübung der betrieblichen Funktionen innerhalb der Teileinheiten. Im Mittelpunkt steht hierbei die Arbeit als zielbezogene menschliche Handlung, aber auch die Ausstattung der Teileinheiten mit den zur Aufgabenerfüllung notwendigen Informationen und Sachmitteln.“[31]

Wie auch schon bei der Gestaltung der Aufbauorganisation liefert das Ergebnis der Aufgabensynthese die Form der Ablauforganisation. Die zentrale Problematik dieser Synthese ist die Festlegung von sogenannten Arbeitsgängen. Eine geschlossene Abfolge von Verrichtungen an einem Objekt wird als ein Arbeitsgang bezeichnet. Nach der Vollendung eines solchen Arbeitsganges wird dieser entweder wiederholt, oder es wird ein anderer Arbeitsgang begonnen. Geht man davon aus, dass Transportvorgänge auch Arbeitsgänge darstellen, so stehen die folgenden vier Probleme im Zentrum der ablauforganisatorischen Gestaltung:

[30] Vgl. Schulte-Zurhausen, 1995, S. 212ff

[31] Schulte-Zurhausen 1995, S. 11

1. Die Erstellung eines Arbeitsganges durch die Beschreibung der an einem Objekt vorzunehmenden Verrichtungen, deren Reihenfolge und zeitliche Verteilung.
2. Die Festlegung der Menge der in einem Zeitraum zu erfüllenden verschiedenen Arbeitsgänge für einen Aufgabenträger
3. Die Bildung von Reihenfolgen, in denen die einzelnen Arbeitsgänge erfüllt werden müssen.
4. Die Festlegung von Pausen zwischen den einzelnen Arbeitsgängen.

Die Techniken der Gestaltung solcher Ablaufprozesse sind im Rahmen dieser Ausarbeitung nicht weiter relevant und werden daher nicht näher betrachtet.

2.4 Das GroupProcess-Projekt

Neben der Klasse der stark strukturierten Prozesse, die von derzeit verfügbaren Workflow-Management-Systemen verarbeitet werden kann, existiert insbesondere in Büroumgebungen von Unternehmen noch eine weitere Klasse von Prozessen, die bisher noch nicht gezielt unterstützt wird: die Klasse der flexiblen, kurzlebigen, schwach strukturierten Ad-hoc-Workflows. In diesem Kapitel wird ein System beschrieben, das auf die speziellen Eigenschaften und Anforderungen von Ad-hoc-Workflows ausgerichtet und dabei in einem aktuellen Workflow-Management-System integriert ist. Zu den wesentlichen Innovationen gehören dabei die gleichzeitige Modellierung und Ausführung von Prozessen. Hinzu kommt die Möglichkeit von partizipativer Modellierung von Workflows und der erleichterte Übergang von schwach strukturierten Prozessen in stärker strukturierte Prozesse. Dieses Konzept soll die einfache Bedienung des Werkzeuges sichern und gleichzeitig die Möglichkeit bieten, stärker strukturierte Abläufe einfach zu generieren.[32]

2.4.1 Konzepte und Ziele

In der o.g. Arbeit über GroupProcess von Dipl.-Inform. Carsten Huth und Professor Dr. Ludwig Nastansky wird davon ausgegangen, dass die beiden Haupttypen des

[32] Vgl. Huth / Nastansky 2000a

Workflow-Managements in einem System unterstützt werden müssen. Dies bedeutet, dass sowohl starre Strukturen im Sinne von Geschäftsprozessautomatisierung als auch flexible, mehr teamgetriebene Abläufe berücksichtigt werden müssen. Als Ansatz für diese Theorie wird das Workflow-Kontinuum von Hilpert und Nastansky herangezogen, welches in [Nastansky / Hilpert 1994] präsentiert wurde.

In aktuellen Workflow-Management-Systemen werden die beiden Phasen Gestaltung und Ausführung (Build-Time und Run-Time) getrennt. Ein Prozessablauf wird damit in einem Modellierungssystem (Vgl. PAVONE ProcessModeler in Kapitel 3.2.2.1) vordefiniert. Der eigentliche Ablauf dieses Prozesses findet dann allerdings in einem Workflow-Runtime-System statt (Vgl. Kapitel 3.2). GroupProcess beschäftigt sich nun mit der Integration der Gestaltung in den Ablauf eines Workflows. Diese Integration ist insbesondere der Gestaltung von Ad-hoc-Workflows zuträglich, da sich deren Ablauf während der Abarbeitung ändern kann. Die folgende Abbildung gibt eine Überblick über die Einordnung von GroupProcess in Workflow-Management-Systeme. Besonders gut verdeutlicht wird die Integration der Modellierungs- und Ausführungskomponenten.

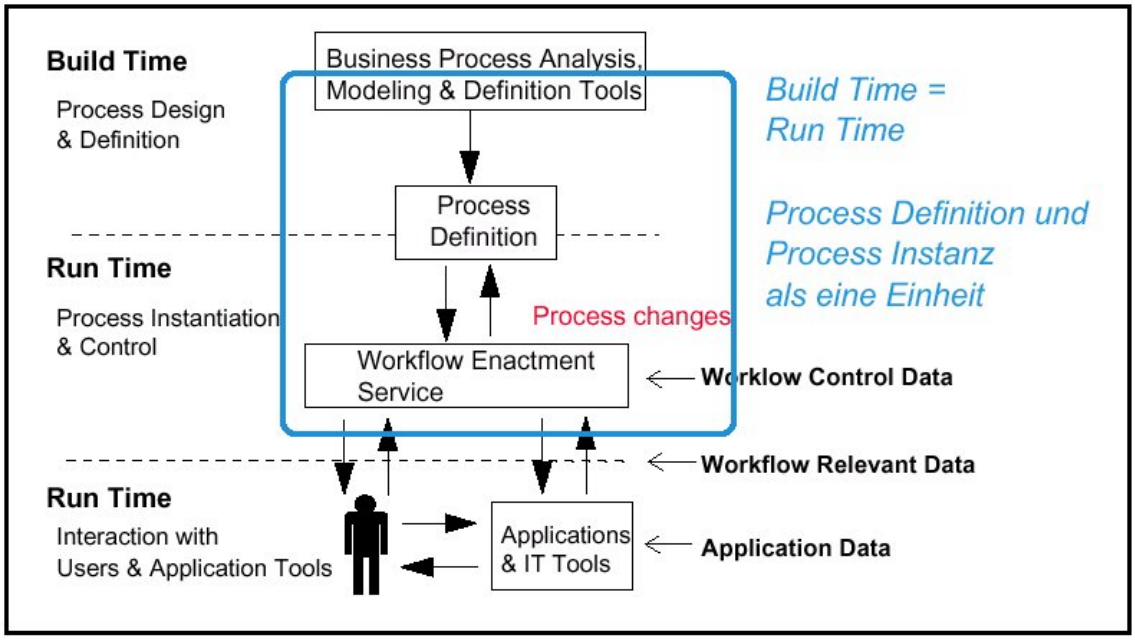

Abbildung 7: Integration von GroupProcess in Workflow-Management-Systeme[33]

[33] Huth / Nastansky 2000b

Demnach hat das GroupProcess-Projekt das Ziel, Konzepte und Werkzeuge zu entwickeln, die auf die o.g. speziellen Eigenschaften von Ad-hoc-Workflows ausgerichtet sind und zugleich in aktuelle Workflow-Management-Systeme wie z.B. PAVONE Enterprise Office integriert werden können. Ein weiteres Ziel ist es, ein solches System verteilt einsetzen zu können, um jedem Bearbeiter die Möglichkeit zu geben, aktiv in den Gestaltungsprozess eines Workflows einzugreifen. Daher wird eine Lösung unter Verwendung des Internets und Lotus Notes-Clients angestrebt.

3 Umgebung und Ressourcen

3.1 Groupwareumgebung Lotus Notes

Im GroupProcess-Projekt wurde Lotus Notes als Groupware-Plattform für die Entwicklung des Ad-hoc-WorkflowModelers eingesetzt. Die Stärken von Lotus Notes liegen unter anderem in der Integration heterogener Datenquellen und Datenformate sowie in verteilten, dokumentenbasierten Datenbanken, die das Konzept des „Information Sharing“ durch Replikation auf betrieblichen LAN- und WAN-Umgebungen realisieren. Durch den konsequenten Abbau proprietärer Standards und Formate zugunsten der Implementierung von offenen Internet-Standards bzw. leistungsstarken Internet-Schnittstellen werden Groupware-Systeme wie Lotus Notes zunehmend zu einer wertvollen Ergänzung und Erweiterung von WWW-basierten Intranetlösungen.

Das wichtigste Merkmal von Lotus Notes ist dessen dokumentenbasierter Ansatz. Ein Dokument in Lotus Notes besteht aus einer Menge Feldern und deren Inhalten. In diesen Feldern können die unterschiedlichsten Daten gehalten werden, wie z.B. Text, Grafiken, Videos, Dateianhänge, Hypertext-Links oder eingebettete Fremdapplikationen wie z.B. Java-Applets[34]. Es handelt sich daher um sogenannte Verbunddokumente, die eine eigene Verarbeitungsintelligenz enthalten können und durch eine „DocumentUniqueID“ eindeutig identifizierbar sind.

[34] Die Möglichkeit des Einbettens von Java-Applets wird auch im Rahmen der vorliegenden Arbeit genutzt.

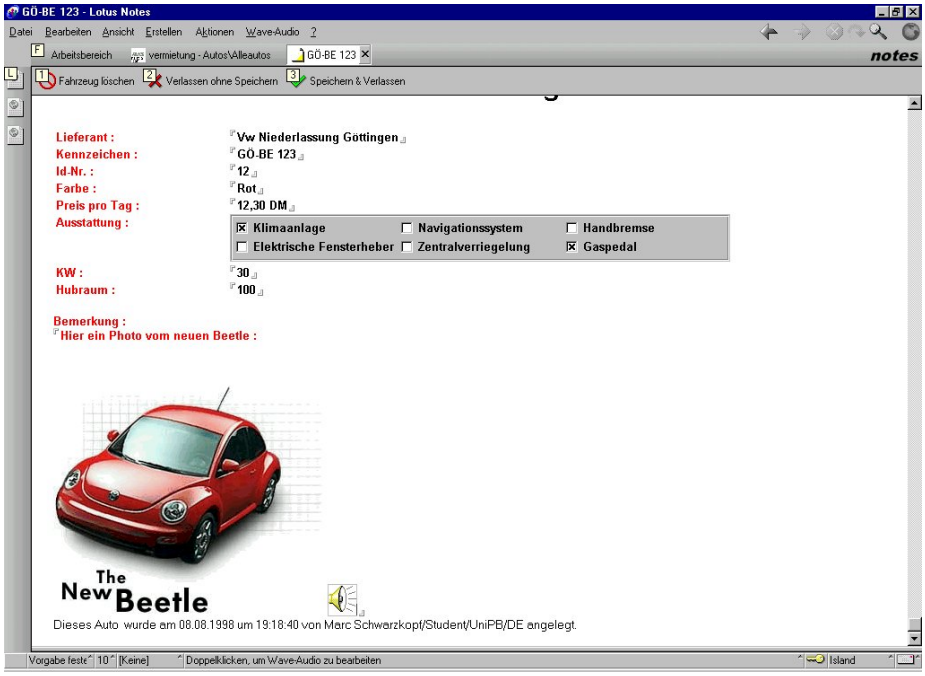

Abbildung 8: Lotus Notes-Verbunddokument [35]

Der hier dargestellte Screenshot gibt ein gutes Beispiel für ein Verbunddokument in Lotus Notes. In diesem Dokument wird nicht nur Text angezeigt, sondern es sind auch Grafiken, Audiodaten und andere „visual components“[36] integriert. Des Weiteren verfügt es im unteren Teil über eine automatisch generierte Zeile mit Verarbeitungslogik, die über das Erstellungsdatum und den Benutzer Aufschluss gibt.

Jedes Dokument kann über unterschiedliche Masken angezeigt werden, so dass je nach Kontext nur eine bestimmte Teilmenge der im Dokument enthaltenen Information angezeigt werden kann. Über Agenten können Aktionen zeit- oder ereignisgesteuert ausgeführt werden.

Lotus Notes-Datenbanken können nicht nur auf den Lotus Notes-Servern (Lotus Domino) abgelegt sein sondern auch lokal beim Benutzer. Der Abgleich zwischen der Server Datenbank und den lokal gehaltenen erfolgt über die Replikation. Bei der Replikation werden lokal gehaltene Datenbanken mit den serverseitigen synchronisiert.

---

[35] Screenshot aus Lotus Notes Praktikum I von Sebastian Scholz & Marc Schwarzkopf 1998

[36] Unter „visual Components“ versteht man diverse Komponenten zur Dialoggestaltung, wie z.B. Button, Checkbox, Choices, ListBox, Radio Buttons ...

Dieses Konzept ist insbesondere für mobile Benutzer interessant, da sie keine ständige Serveranbindung benötigen.

3.2 Workflow-Management-System PAVONE Enterprise Office

Die 1994 gegründete PAVONE AG hat ihren Hauptsitz in Paderborn, Deutschland und weitere Niederlassungen in England und den USA. Mittlerweile ist PAVONE einer der führenden Anbieter innovativer, branchenunabhängiger Informations-managementlösungen auf Basis der Groupware-Plattform Lotus Notes / Lotus Domino.

Neben internetfähigen Standardwerkzeugen für Projekt- und Knowledge-Management sowie Business Intelligence entwickelt die PAVONE AG das Workflow-Management-System Enterprise Office.

PAVONE Enterprise Office ist eine gruppenbasierte Lotus Domino Lösung für das Adress-, Korrespondenz- und Dokumentenmanagement im Bürobereich. Dieses Groupware-System verknüpft ausgereiftes Office- und Dokumentenmanagement mit innovativem Workflow- und Knowledge-Management.[37]

Das PAVONE Enterprise Office Projekt setzt sich aus vier Datenbanken zusammen, die jeweils mit mitgelieferten Modellierungswerkzeugen zu bearbeiten sind. Zu diesen Datenbanken zählt die Organisation Database, in der die Aufbaustruktur einer Organisation abgelegt wird. In der Process Database werden die Informationen über sämtliche Prozesse gehalten sowohl über die vordefinierten Prozesse wie auch über spontane (Ad hoc) Abläufe.

Die Office Database bildet die Kommunikationsbasis für sämtliche Bearbeitungen innerhalb von Enterprise Office. In ihr werden neue Prozesse gestartet, Berichte geschrieben, E-Mails verfasst, Adressen verwaltet usw. Hier treten auch die Ereignisse auf, die zum Weiterleiten eines Dokuments nötig sind, wie z.B. die Abarbeitung eines Dokumentes durch einen Bearbeiter. In diesem Fall muss das Dokument an den nächsten Bearbeiter weitergeleitet werden.

[37] Vgl. http://www.pavone.de

In der Settings Database werden Einstellungen gespeichert, die zur Ausführung der Office Database nötig sind, wie z.B. Konfigurationseinstellungen und Vorlagen.

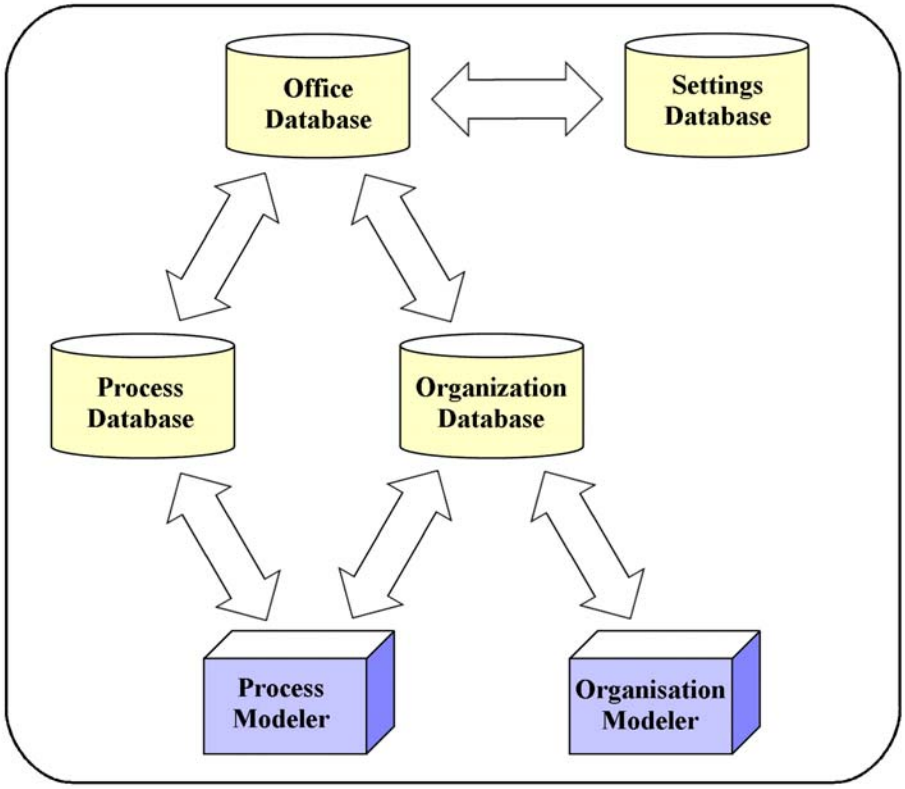

Abbildung 9: Zusammenhang der PAVONE Datenbanken

Im Rahmen dieser Arbeit sollen die Teile Enterprise Office Organisation Database, Enterprise Office vordefinierte Workflows und Enterprise Office Ad-hoc-Workflows des Produktes Enterprise Office angeführt werden, da Teile dieser Strukturen im fünften Kapitel bei der Vorstellung des Organizationchart- und des WorkgroupViewers verwendet werden.

3.2.1 Enterprise Office Organisation Database

Im Kontext des Workflow-Managements bietet die Organisation Database von PAVONE Enterprise Office die Basis für die Modellierung der innerbetrieblichen Struktur. In dieser Datenbank werden alle die Aufbauorganisation betreffenden Daten festgehalten. PAVONE stellt ein Modellierungswerkzeug zur Verfügung, mit dem diese

Aufbaustruktur komfortabel gestaltet werden kann. Die wichtigsten Elemente; Personen, Abteilungen, Arbeitsgruppen und Rollen werden im Folgenden näher beschrieben.

3.2.1.1 Personen

Hier werden die Daten aller natürlichen Personen einer Organisation abgelegt. Neben den persönlichen Daten zu den einzelnen Personen werden Informationen zu Abteilungs- und Arbeitsgruppenzugehörigkeit sowie Kostenstrukturen gehalten und Auslastungsgrade der Personen ermittelt.

3.2.1.2 Sachen

Unter einer Sache wird im Allgemeinen eine Ressource verstanden. Unter „Sachen“ in PAVONE Enterprise Office können sämtliche Details zu vorhandenen Ressourcen abgelegt werden. Praktisch gesehen sind die Details, die „Sachen“ zugeordnet werden können, gleich denen bei den Personen.

3.2.1.3 Abteilungen

In diesem Bereich wird die gesamte Hierarchie der Organisation über die Einordnung der Abteilungen in die Organisationsstruktur festgelegt. Zu jeder Abteilung können der Abteilungsleiter, der Stellvertreter und die Mitglieder durch Auswahl aus den angelegten Personen zusammengestellt werden. Mit dem von PAVONE angebotenen OrganizationModeler können die Strukturen der Unternehmung auch graphisch aufgebaut werden.

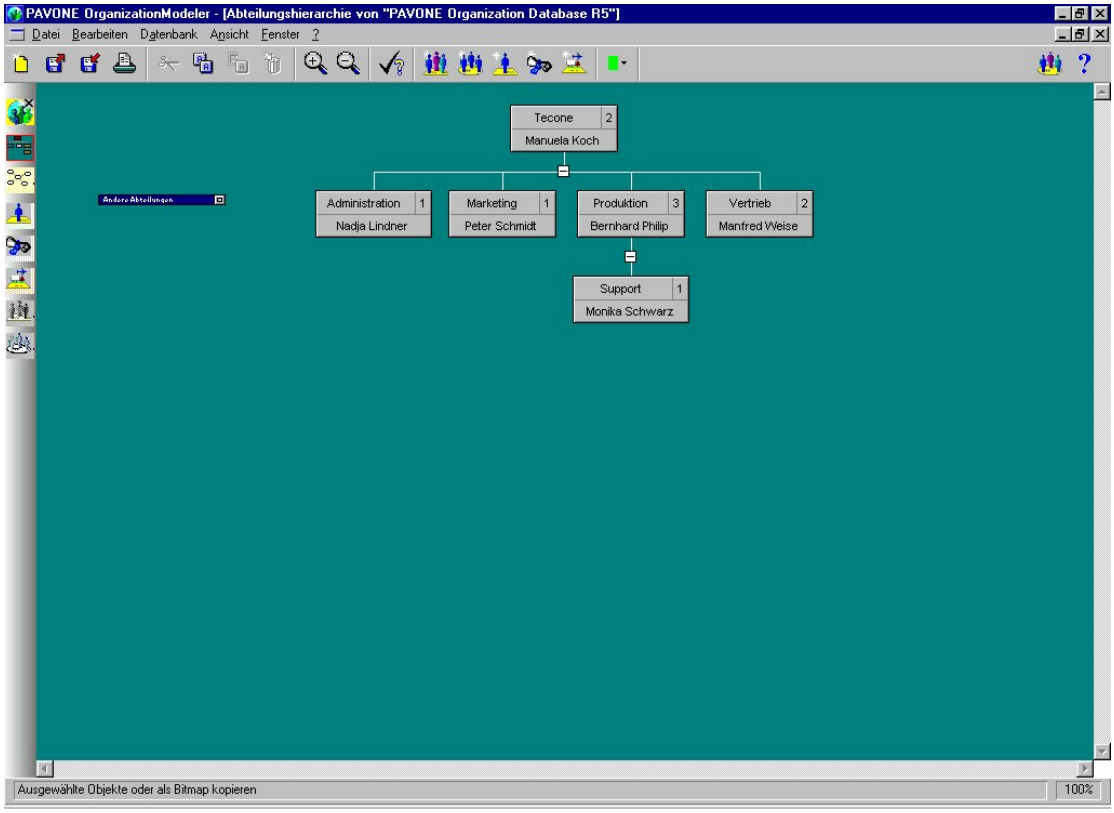

Abbildung 10: Die Darstellung der Abteilungen in PAVONE[38]

3.2.1.4 Arbeitsgruppen

Arbeitsgruppen können in PAVONE Enterprise Office ähnlich gesehen werden wie die Abteilungen, wobei allerdings bei den Arbeitsgruppen keine Hierarchie existiert (siehe auch Kapitel 2.3.2.3). Arbeitsgruppen sind so gesehen losgelöste Teams, die sich aus Mitarbeitern der Abteilungen zusammensetzen und für temporäre Projekte eingesetzt werden. Ein weiterer Unterschied zu den Abteilungen besteht darin, dass Personen mehreren Arbeitsgruppen zugeordnet werden können. Auch für die Arbeitsgruppen bietet PAVONE ein Werkzeug an, wodurch der Benutzer bei der Modellierung unterstützt wird.

[38] Screenshot des PAVONE OrganizationModeler

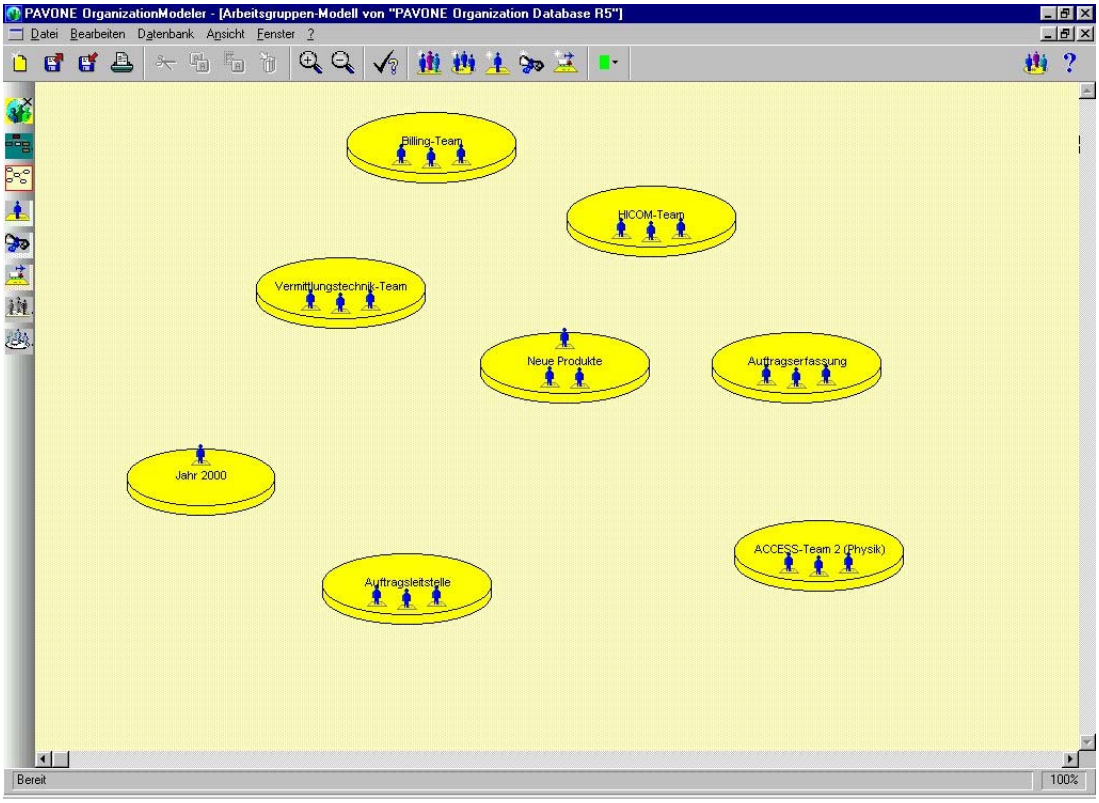

Abbildung 11: Die Darstellung der Arbeitsgruppen in PAVONE[39]

3.2.1.5 Rollen

Durch Rollen werden Fähigkeiten beschrieben, eine bestimmte Tätigkeit ausführen zu können oder eine bestimmte Qualifikation zu haben. Diese Rollen können Personen, Abteilungen oder Arbeitsgruppen zugeordnet werden. Dadurch ergibt sich im Kontext des Workflow-Managements die Möglichkeit der flexiblen Gestaltung, da es die Möglichkeit schafft, eine Aufgabe an eine bestimmte Fähigkeit zu knüpfen, ohne dass die dahinterstehende Instanz bekannt sein muss.

3.2.2 Enterprise Office Workflows

In der PAVONE Process Database werden die mit dem PAVONE ProcessModeler erstellten Prozessdefinitionen abgelegt. Sobald ein neuer Prozess initiiert wird, wird auf diese Prozessdefinitionen von der PAVONE Office Database aus zugegriffen, indem die

[39] Screenshot aus PAVONE Enterprise Office

Prozessdefinitionen von der PAVONE Process Engine ausgelesen und in der Anwendungsdatenbank PAVONE Office Database ausgeführt werden. Die PAVONE Office Database bezieht aus dieser Datenbank die Informationen zur Steuerung des Prozessablaufs.[40]

3.2.2.1 Vordefinierte Workflows

Für vordefinierte Workflows stellt PAVONE ein umfangreiches Entwicklungswerkzeug, den ProcessModeler zur Verfügung. Damit können immer wiederkehrende, gut strukturierte Arbeitsabläufe für die dauerhafte Nutzung innerhalb der PAVONE Office Database zusammengestellt werden.

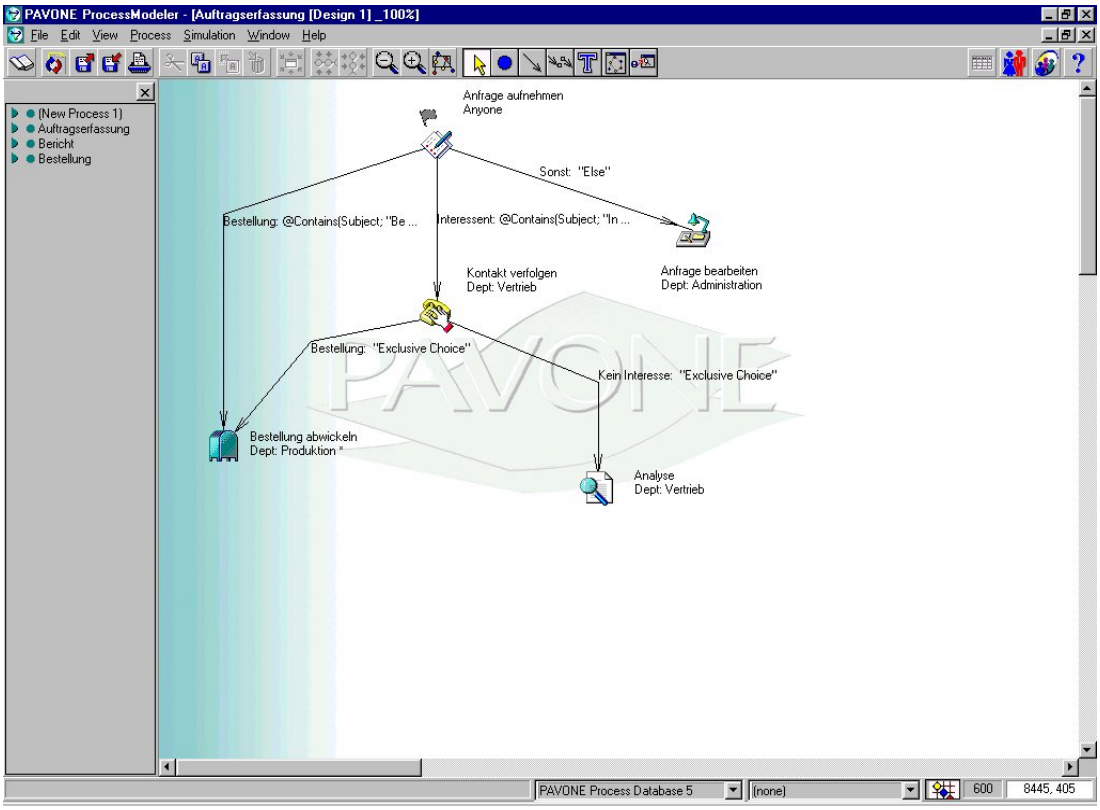

Abbildung 12: Der PAVONE ProcessModeler [41]

Für die Modellierung eines neuen Prozesses mit dem PAVONE ProcessModeler kann auf existierende Prozessdefinitions-Datenbanken zurückgegriffen werden, die so als

[40] Vgl. PAVONE Enterprise Office Hilfe – „Funktion der PAVONE Process Database“

[41] Screenshot des PAVONE ProcessModelers

Baustein-Bibliotheken dienen. Auf ihrer Basis können dann im PAVONE ProcessModeler neue Strukturen erstellt werden.

Bei dem ProcessModeler handelt es sich um ein externes Tool, mit dem die einzelnen Workflows graphisch gestaltet werden können. Diese Anwendung ist unabhängig von Lotus Notes und benötigt zu ihrer Ausführung lediglich die Organization Database und die Process Database von PAVONE. Zusätzlich gibt es innerhalb dieses Werkzeuges die Möglichkeit, die kreierten Workflows mittels der integrierten Simulationsunterstützung simulieren zu können. Dabei können einerseits für jeden Durchlaufknoten die Zeitpunkte der Ankunft und der Weiterleitung der Dokumente betrachtet, und andererseits diese durch verschiedene Maßnahmen optimiert werden. Außerdem werden die durch die Bearbeitungs- und Weiterleitungszeiten entstandenen Kosten errechnet.

3.2.2.2 Ad-hoc-Workflows

Die mit PAVONE Enterprise Office gelieferte Lösung zur Abwicklung von Ad-hoc-Workflows bietet auch die Möglichkeit, spontane Abläufe zum Zeitpunkt ihres Entstehens zu modellieren. Dazu ist in der Office Database ein Werkzeug integriert, welches bei Bedarf aufgerufen wird. Die Gestaltung eines Ad-hoc-Workflows verläuft innerhalb von zwei Masken :

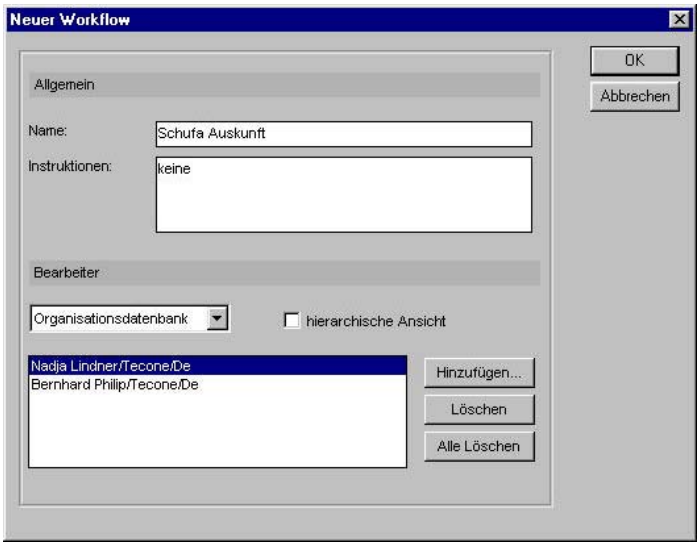

Abbildung 13: Details eines Ad-hoc-Workflows in PAVONE Enterprise Office[42]

[42] Screenshot des Ad-hoc-Workflow Modelers aus der PAVONE Enterprise Office Hilfe

Diese Abbildung zeigt die Details zu einem Prozessschritt innerhalb des Ad-hoc-Workflows. Hier sind der Bearbeiter und dessen Aufgabe einzustellen. Nach der Bestätigung wird der ausgewählte Bearbeiter und die dazugehörige Aufgabe im folgenden Workflow-Übersichtsfenster eingetragen.

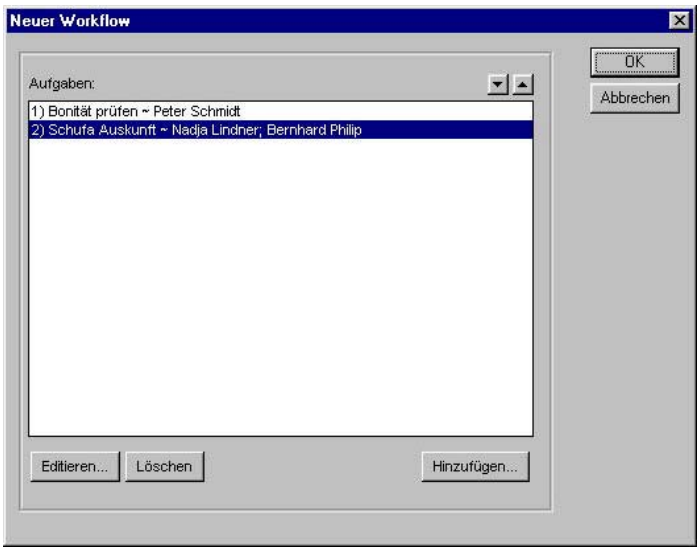

Abbildung 14: Übersicht über den Workflow bei PAVONE Enterprise Office[43]

In der oben gezeigten Abbildung erkennt man, dass bei diesem Typ des Workflows nur lineare Prozessabläufe generierbar sind. Im Falle eines komplizierteren Ablaufes, z.B. mit einer Verzweigung an einer bestimmten Stelle auf parallel ablaufende Aufgaben und einer späteren Zusammenführung der Teilzweige, wäre der Aufruf des externen ProcessModelers nötig.

3.2.3 Umgebung und Ressourcen von GroupProcess

Als Groupware Plattform entschieden sich die Entwickler des GroupProcess-Projektes für das schon recht weit verbreitete Lotus Notes- / Lotus Domino-Konzept. Ein Vorteil bei der Wahl dieses Systems zur Realisation von GroupProcess ist die mögliche Integration der Java-Programmiersprache, auf deren Vorteile im Kapitel 3.3 näher eingegangen wird.

[43] Screenshot aus PAVONE Enterprise Office

Aufgrund der in 2.4.1 genannten Zielkriterien zur Erstellung eines OrganizationchartViewers und eines WorkgroupViewers muss eine Lösung entweder auf der Java-Technologie basieren oder auf der Einbeziehung von Lotus Script. Lotus Script erscheint erstmalig ab Version 4 in Lotus Notes. Mit dieser Scriptsprache ist es möglich, komplexere Anforderungen an Groupware-Systeme zu bewältigen. Allerdings fehlt dieser Scriptsprache eine graphische API[44] zur Generierung von Benutzungsschnittstellen und graphischen Ausgaben.

Im Bereich der Integration von Java bietet Lotus Notes die Möglichkeit der Einbindung von Java-Applets in Dokumente. Bei Einsatz der Java-Technolgie ist es dem Entwickler möglich, auch graphische Oberflächen zu modellieren, da Java eine dementsprechende API bereitstellt. Des Weiteren ist aufgrund der Plattformunabhängigkeit von Java der Einsatz eines solchen Systems in beliebigen Umgebungen gewährleistet. Ein solches Applet ist daher sowohl direkt auf einem Lotus Notes-Client lauffähig als auch in einem Internetbrowser im Rahmen eines Web-Interfaces.

Aus den genannten Gründen wurde das Ad-hoc-Workflow-Modellierungswerkzeug in Java implementiert. Die folgende Abbildung zeigt eine aktuelle Version des Ad-hoc-WorkflowModelers

[44] API – Application Programing Interface

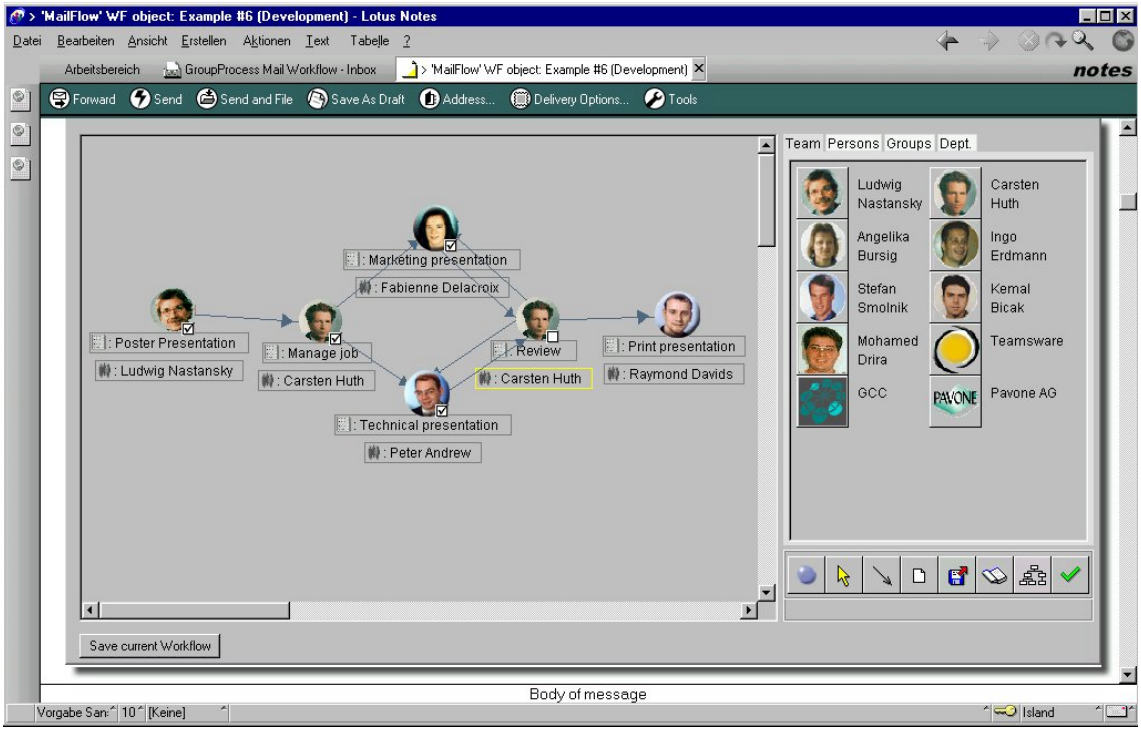

Abbildung 15: GroupProcess - der Ad-hoc-WorkflowModeler[45]

Das oben gezeigte Werkzeug ist direkt aus einer Lotus Notes-Datenbank aufrufbar. Mit ihm ist es möglich, mit wenig Interaktion einen Ablauf zu generieren. Die in den Knoten des entwickelten Workflows aufgeführten Personen, Arbeitsgruppen und Abteilungen einer Organisation können im rechten Bereich des Applets ausgewählt werden. Die dort angezeigten Daten sowie die Icons stammen aus der PAVONE Organization Database. Im linken Bereich wird der aktuell entwickelte Workflow visualisiert. Durch einfaches Anwählen eines Symbols auf der rechten Seite mit der Maus wird ein neuer Knotenpunkt mit den Daten des entsprechenden Symbols erzeugt. Im Verknüpfungsmodus werden die Verbindungen zwischen den erzeugten Punkten durch einfaches Ziehen mit der Maus erstellt. Im Verschiebemodus können die Knoten durch die Maus im Anzeigebereich beliebig verschoben werden, um eine optimale Darstellung der Verbindungen des Ablaufes zu erreichen.

[45] Screnshot des Ad-hoc-Workflow-Modelers des GroupProcess-Projektes in der zum Zeitpunkt der Erstellung dieser Arbeit aktuellen Version.

Weitere Funktionen des Werkzeuges sind: Löschen des erzeugten Ablaufes, Speichern des Ablaufs und Anzeigen der Historie.

3.3 Java

Java wurde von der Firma Sun Microsystems entwickelt und erstmals am 23.Mai 1995 als neue, objekt-orientierte, einfache und plattformunabhängige[46] Programmiersprache vorgestellt. Sun Microsystems besitzt das Schutzrecht auf den Namen Java und stellt damit sicher, dass nur „100% richtiges Java“ diesen Namen tragen darf. Die Sprache Java ist im Allgemeinen kostenlos für annähernd alle Computersysteme verfügbar.

Java geht auf die Sprache Oak zurück, die 1991 im sogenannten Green-Projekt mit dem Ziel entwickelt wurde, eine einfache und plattformunabhängige Programmiersprache zu schaffen, mit der nicht nur Computer wie UNIX-Workstations, PCs und Apples programmiert werden können, sondern auch die in Haushaltsgeräte eingebauten Micro-Computer, wie z.B. in Videorecordern, Autos, Kreditkarten und Sicherheitssystemen und vor allem auch in TV-Settop-Boxen für „intelligente“ Fernsehapparate.

Allgemein anerkannt wurde Java aber erst seit 1996 in der Verbindung mit Internet-Anwendungen und Web-Browsern sowie mit der Idee eines NC (Network Computer), der im Gegensatz zum PC (Personal Computer) nicht lokal installierte, maschinenspezifische Software-Programme benötigt, sondern die Software in Form von Java-Klassen dynamisch über das Netz (Inter- / Intranet) von einem zentralen Server laden kann. Diese grundlegende Idee wurde später zu einem allgemeinen „Application Service Providing“ (ASP) erweitert.[47]

Aus diesem Grund wird Java vielfach als reine Internetsprache bezeichnet, was in diesem Sinne nicht stimmt, da Java ursprünglich gar nicht auf das Internet zielte. Außerdem drückt diese Bezeichnung nicht das eigentliche Potential aus, welches hinter Java steckt.

Bei Java-Programmen müssen zwei grundsätzliche Arten unterschieden werden :

---

[46] Siehe Kapitel 3.3.1

[47] Vgl. Partl

Applikationen (applications) : Java-Applikationen sind Computer-Programme mit dem vollen Funktionsumfang, wie er auch bei anderen Programmiersprachen gegeben ist. Applikationen können als lokale Programme auf dem Rechner des Benutzers laufen oder als Client-Server-Systeme über das Internet bzw. über ein Intranet oder als Server-Programme auf einem Web-Server. Der entscheidende Vorteil zu Applikationen, die in anderen Programmiersprachen entwickelt worden sind, ist, diese Java-Applikationen auf unterschiedlichen Plattformen zum Einsatz bringen zu können. Ein Nachteil ist die langsamere Ausführungsgeschwindigkeit von Java-Programmen (siehe auch Kapitel 3.3.1)

Applets : Java-Applets werden innerhalb einer Web-Page dargestellt und unter der Kontrolle eines Web-Browsers ausgeführt. Sie werden meist über das Internet von einem Server geladen, und spezielle Sicherungen innerhalb des Web-Browsers ("Sandkasten", sandbox siehe Kapitel 3.3.2) sorgen dafür, dass sie keine unerwünschten Wirkungen auf den Client-Rechner haben können. So können Applets z.B. im Allgemeinen nicht auf lokale Files, Systemkomponenten oder Programme zugreifen und auch nicht auf Internet-Verbindungen außer zu dem einen Server, von dem sie geladen wurden.[48]

Einige grundsätzliche Konzepte der Java-Technologie sollen hier im Folgenden zum besseren Verständnis dargestellt werden:

3.3.1 Plattformunabhängigkeit

Wie bereits in der Einleitung dieses Kapitels erwähnt, handelt es sich bei Java um eine plattformunabhängige Programmiersprache. Plattformunabhängigkeit bedeutet, dass ein und dasselbe Programm ohne weitere Anpassungen direkt auf unterschiedlichen Systemen und somit auch Betriebssystemen eingesetzt werden kann.

Diese Plattformunabhängigkeit wurde dadurch erreicht, dass die Programme nicht direkt auf den unterschiedlichen Systemen laufen sondern auf einer sogenannten „Virtual Machine“(VM). Diese Virtual Machine kann als Adapterprogramm verstanden werden, welches auf der einen Seite die Schnittstelle zu Java bietet und auf der anderen Seite die Verbindung zu dem jeweiligen Betriebssystem herstellt. Für jedes existierende

[48] Vgl. Partl

Betriebssystem, auf dem Java-Programme lauffähig sein sollen, muss eine entsprechende Version dieser VM existieren, die an den jeweiligen Maschinencode des Systems angepasst ist.

Java-Programme werden allerdings vom Entwickler nicht in dem für die Java Virtual Machine notwendigen Code erzeugt, sondern als Java-Quellcode. Dieser Java-Quellcode ist praktisch eine Ansammlung von Java-Befehlen, die allerdings in einer dem Menschen verständlicheren Form vorliegen. Diese Befehle sind von der VM nicht direkt in die Maschinensprache des entsprechenden Betriebssystems übersetzbar. Als Zwischenschritt ist es also notwendig, den Java-Quellcode in eine für die Virtual Machine lesbare Sprache zu übersetzen: den Java-Bytecode. Deshalb wird mit der Java-Entwicklungsumgebung ein Java-Compiler geliefert, der diese Aufgabe übernimmt.

Wenn der Java-Quellcode in Java-Bytecode übersetzt (kompiliert) wurde, ist das dahinterstehende Programm transportfähig und auf jedem Betriebssystem, welches eine Java Virtual Machine besitzt, ausführbar.

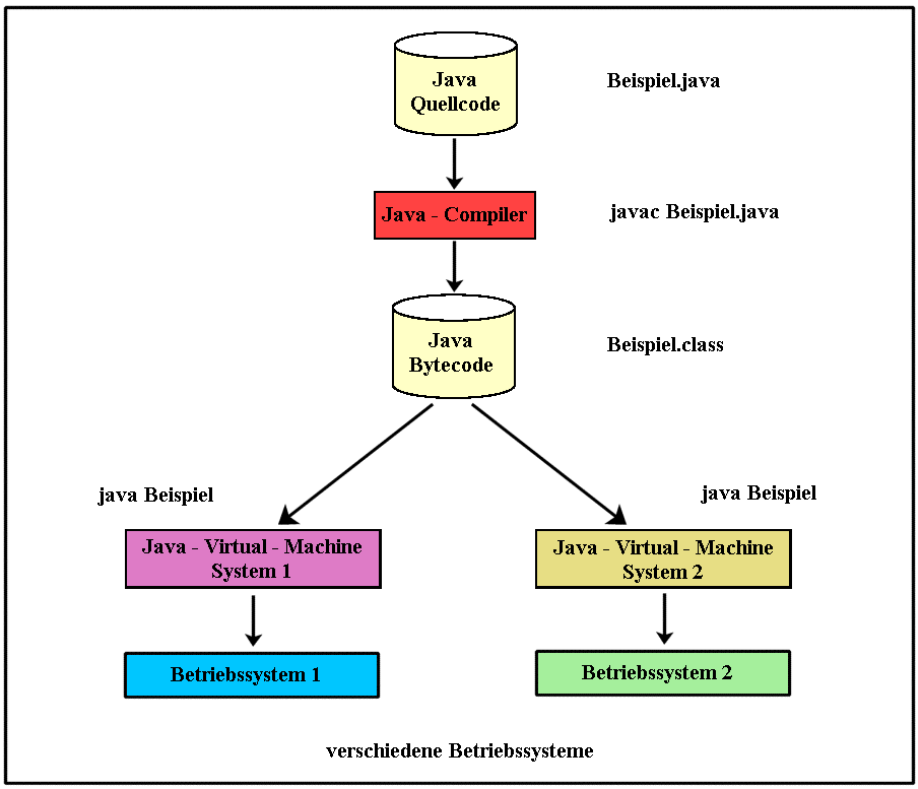

Abbildung 16: Die Virtual Machine von Java

Da allerdings die ausführbaren Java-Programme im Java-Bytecode auf dem System vorliegen und nicht in dem auf dem jeweiligen System üblichen Maschinencode, ist natürlich zur Ausführung eines solchen Programms zusätzlich zum eigentlichen Rechenaufwand des Programms noch Rechenaufwand zum Übersetzen des Java-Bytecodes notwendig, was sich auf die Performance des Programms negativ auswirken kann. Aufgrund der immer weiter zunehmenden Leistung heutiger Prozessoren wird allerdings dieser Zusatzaufwand für die Interpretation mehr und mehr vernachlässigbar.

3.3.2 Sicherheitskonzepte

Aufgrund der Möglichkeit, Java-Programme über das Internet zu verteilen, ist es notwendig, Sicherheitskonzepte zu haben, die den Client, den Server und die Verbindung zwischen den beiden Systemen gegen unerwünschte Zugriffe schützen. Ein Teil dieser Sicherheitskonzepte, die im Laufe der Weiterentwicklung von Java entstanden, werden hier dargestellt :

3.3.2.1 Sandbox

Das Sandbox-Konzept war das Sicherheitskonzept der ersten Stunde. Es wurde zum erstenmal in die Java-Version 1.0 integriert und basiert auf der Annahme, dass jeglicher bereits auf dem System installierte Java-Code sicher ist und jeglicher Code, der von außen kommt, unsicher. Vor diesem, von außen kommenden Code werden alle Daten und Ressourcen vor Manipulation und Weitergabe durch die folgenden Restriktionen geschützt:

- Keine Zugriffe auf das Dateisystem
- Kein Zugriff auf Systemresourcen wie Zwischenablage, Drucker und Benutzer-Systemdaten.
- Keine Netzwerkverbindungen mit anderen Rechnern, mit Ausnahme des Ursprungs-Host des Programms.
- Kein Start von weiteren Prozessen oder Programmen.
- Keine Beendigung der Java Virtual Machine.
- Kein Laden dynamischer Bibliotheken (z.B. C++ Code)

- Keine Veränderung von Sicherheitseigenschaften.
- Keine Überladung der bereits vorhandenen Java-Systemklassen durch andere Versionen

Diese Maßnahmen sind allerdings für bestimmte Anwendungen zu restriktiv und verhindern teilweise den Einsatz von Java-Applets in Intranets, in denen eine gelockerte Sicherheitsstufe erwünscht ist.[49]

Daher wurde ab der Java-Version 1.1 die Möglichkeit der im Folgenden erläuterten signierten Applets geschaffen.

3.3.2.2 Signierte Applets

Durch die Schaffung signierter Applets ist ab der Java-Version 1.1 die Möglichkeit gegeben, auch Java-Code von anderen Quellen, abgesehen von den lokal im System installierten, Zugriffe auf sensible Systemressourcen zu gestatten. Es muss in diesem Zusammenhang sichergestellt werden, dass der am Client empfangene Java-Code auch tatsächlich von dem Server kommt, bei dem er angefragt wurde. Zu diesem Zweck werden sämtliche zu übertragende Klassen vom Server in ein Archiv gepackt und mit Hilfe der dafür eingeführten Java-Krypto-Klassen digital signiert. Die Problematiken bei diesem System liegen darin, dass solche Signaturen kostenintensiv zu zertifizieren sind, Browserhersteller unterschiedliche Formate zur Übertragung der Klassen fordern und die Anwendung des Prinzips auf Applets beschränkt ist.

3.3.2.3 Protection Domains

Ab Java-Version 2 stehen die sogenannten Protection Domains zur Verfügung, die es dem Systemadministrator erlauben, für bestimmte Internetdomains Rechte zum Download von Java-Programmen zu erteilen. Dies ist vergleichbar mit der Zugriffsrechtevergabe in Betriebssystemen wie UNIX oder Windows NT, mit dem Unterschied allerdings, dass es sich in diesem Fall nicht um Benutzer handelt, sondern um Internetdomains. Die vom Administrator vorgegebenen Rechte werden lokal auf dem ausführenden System in einer sogenannten Policy-Datenbank gespeichert und von den Browsern ausgelesen.

3.3.3 Objektorientiertheit

Eine zentrale Eigenschaft von Java ist die Objektorientiertheit. Diese Eigenschaft bedeutet, dass im Gegensatz zu prozedurorientierten Programmiersprachen die Trennung von Datenebene und Anweisungsebene aufgehoben wird. Zusammengehörende Anweisungen bilden eine zusammengehörende, abgeschlossene und eigenständige Einheit: das Objekt. Objekte bestehen also aus zwei Bestandteilen, den Attributen und den Objektmethoden.

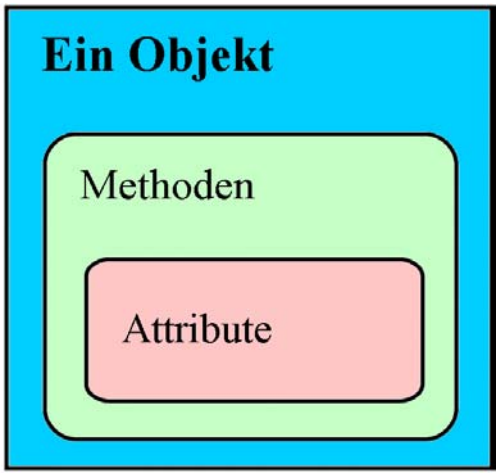

Abbildung 17: Ein Objekt

Objekte sind nach außen nur durch ihre Schnittstellen und Methoden definiert. Dadurch wird die komplette innere Struktur eines Objekts vor anderen verborgen (Information Hiding und Datenkapsel). Die Objektmethoden stellen praktisch einen Programmcode dar, mit dem die im Objekt gehaltenen Daten modifiziert bzw. abgerufen werden können.

Die Baupläne für Objekte werden Klassen genannt. Objekte, die als Rahmen eine Klasse nutzen, sind eine Instanz dieser Klasse. Objekte einer Klasse stellen die selben Methoden bereit und besitzen die gleiche Datenstruktur. Sie unterscheiden sich nur durch die Ausprägungen der in ihnen gehaltenen Daten. Objekte einer Klasse, die identische Daten enthalten, werden dennoch unterschieden.

Von diesen Bauplänen für Objekte können andere Klassen profitieren, indem sie Teile der Strukturen für ihre eigenen Konstrukte nutzen. Dieser Vorgang wird als *Vererbung*

[49] Vgl. Java Spektrum 1999

bezeichnet. Eine Klasse, die von einer anderen Klasse erbt, wird als Subklasse bezeichnet, die vererbende Klasse als Superklasse. Im Gegensatz zu anderen Programmiersprachen wie z.B. C++, ist unter Java die sogenannte multiple Vererbung nicht möglich, was bedeutet, dass eine Klasse nur Konstrukte von genau einer Superklasse erben kann.

Schnittstellen (Interfaces) können genutzt werden, um einer bestimmten Klasse von Objekten vorzuschreiben, welche Methoden implementiert werden müssen. Jede Klasse, die ein solches Interface implementiert, muss die in der Interfaceklasse definierten Methoden bereitstellen. Interfaces sind eine wichtige Möglichkeit zur Kommunikation zwischen Java-Klassen und werden daher noch einmal im Kapitel 3.6.2 weiter erörtert.

Bei komplexeren Anwendungen kann eine unübersichtliche Klassenstruktur entstehen, die dem Programmierer die Implementation des Gesamtprojektes erschwert. Daher ist es unabdingbar, ein System einzusetzen, welches das Gesamtprojekt überschaubar gestaltet. Dazu existieren unter Java sogenannte Packages (Pakete), in denen Klassen und Interfaces ähnlicher Zugehörigkeit zusammengefasst werden können. Diese Packages müssen mit der Struktur des Verzeichnisbaums übereinstimmen, da Java bei der Übersetzung des Quellcodes die Hierarchie der Packages mit der des Verzeichnisbaums vergleicht.

3.4 Techniken der Benutzungsschnittstellen- und Dialoggestaltung

Die im Rahmen dieser Arbeit zu erstellenden Werkzeuge zur Visualisierung von Abteilungs- und Gruppenstrukturen innerhalb von Organisationen bedürfen der Interaktion zwischen den Benutzern der Systeme und den Systemen selbst. Wie aus der Beschreibung des Kontinuums der Benutzertypen in Kapitel 4.1 hervorgehen wird, werden solche Werkzeuge von Benutzern aus den unterschiedlichsten Bereichen eines Unternehmens sowie sogar von organisationsexternen Benutzern angewandt. Aus diesem Grund werden die Aspekte der Mensch-Rechner-Interaktion (HCI[50]) im folgenden theoretisch erörtert werden.

[50] HCI – „Human Computer Interaction“

3.4.1 Benutzungsschnittstellen

Graphische Benutzungsschnittstellen, mit der Maus als wesentlichem Eingabegerät, erleichtern die Mensch-Rechner-Interaktion erheblich. In der Informatik hat sich der Begriff User Interface[51] (UI) anstelle der eigentlich korrekten Bezeichnung Benutzungsschnittstelle eingebürgert.[52]

Seit der Einführung von EDV-Systemen in der Arbeitswelt stellt sich die Frage nach menschgerechten Softwareprodukten. Seit ca. 20 Jahren versucht die Software-Ergonomie, die Wissenschaft von der Anpassung von Software an den Menschen, eine Antwort auf diese Frage zu geben.

> „Die Software-Ergonomie hat sich zur Aufgabe gemacht, die Merkmale benutzer- und aufgabengerechter Software zu erforschen und konstruktive Verfahren sowie Softwareunterstützung für den Prozess der Gestaltung von Benutzungsschnittstellen zu entwickeln. Ihr zentrales Anliegen ist die Optimierung des Zusammenspiels aller Komponenten der Arbeitssituation von Computerbenutzern: Mensch, Aufgabe, Technik und organisatorischer Rahmen.“ [53]

Benutzungsschnittstellen bestehen aus Softwaremodulen, die zum einen die Oberflächen und zum anderen die Interaktion mit dem Benutzer bereitstellen. Die nachfolgende Abbildung zeigt den Zusammenhang zwischen diesen Modulen und Anwendungssystemen dar.

[51] Übersetzt „Benutzer Schnittstelle“ statt „Benutzungsschnittstelle“

[52] Vgl. Steinmetz 1993, S. 401

[53] Maaß et al. 1993

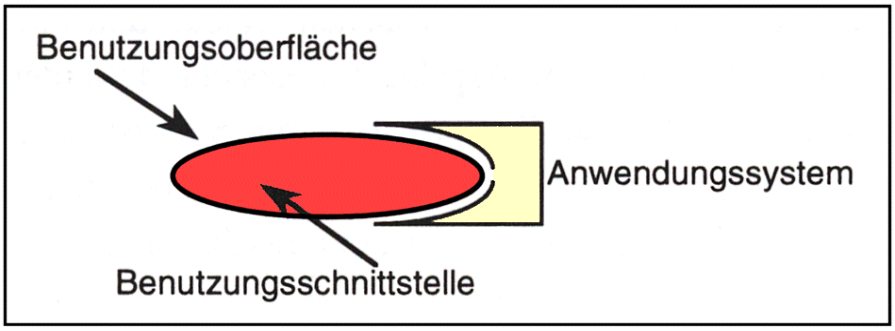

Abbildung 18: Benutzungsoberfläche und Benutzungsschnittstelle in Kombination mit Anwendungssystemen[54]

Um Benutzungsoberflächen menschgerecht zu gestalten, steht die Forderung nach Benutzerfreundlichkeit an erster Stelle. Um dies zu erreichen, wird Benutzerfreundlichkeit in der Literatur durch die folgenden Punkte definiert:

Leicht erlernbare Bedienung: Die Bedienung der Applikation muss für die verschiedenen Benutzer einfach zu erlernen sein. Dies wird im Allgemeinen dadurch erreicht, dass die Gestaltung der Oberflächen der Applikation an die Schnittstellen anderer im System befindlicher Applikationen angepasst wird.

Einprägsame Bedienungsregeln: Die Regeln für die Bedienung der Applikation sollen so einfach gehalten werden, dass sie vom Anwender leicht im Gedächtnis zu behalten sind. Durch intuitive Assoziation mit Bekanntem soll dem Benutzer die Möglichkeit gegeben werden, die Anwendung spontan bedienen zu können. Somit sollte z.B. ein Doppelklick auf eine Zeile einer Datenauflistung das Öffnen des kompletten Datensatz bewirken.

Effektive Bedienbarkeit: Anwendungen mit effektiver Bedienbarkeit zeichnen sich durch die folgenden fünf Merkmale aus:

- Logisch zusammenhängende Funktionen sind gruppiert, und die Bedienung ist gleichartig gestaltet.
- Statt textueller Ein- und Ausgabe werden graphische Symbole verwandt.

[54] aus Daldrup 1995,S. 4

- Zwischen unterschiedlichen Anwendungen können Daten in verschiedenen Medien ausgetauscht werden.
- Hauptfunktionen sind direkt, ohne unnötige Menüverzweigung ansprechbar.
- Die Bedürfnisse sämtlicher Benutzergruppen werden abgedeckt.

Kontextsensitive Hilfefunktionen: Beim Einbetten der Hilfefunktionen sollte darauf geachtet werden, dass jeweils die Hilfe zu der gerade benutzten Funktion oder Ein- / Ausgabe abgerufen werden kann. Ein Rückgriff auf die Bedienungsanleitung sollte vermieden werden können.

Ästhetik: Unter der Ästhetik einer graphischen Oberfläche versteht man die in ihr angewandte Kombination von Schriftart, Farben, Auflösung und Form. Diese Kriterien sind über alle Oberflächen einer Anwendung gleich zu wählen und sollten innerhalb einer Oberfläche nicht zu sehr variieren.[55]

3.4.2 Dialoggestaltung und -steuerung

Ein Dialog, im Gegensatz zu einem Monolog, beschreibt eine Konversation zwischen zwei oder mehreren Parteien. Beim Design von Benutzeroberflächen hat der Dialog eine spezifischere Bedeutung, nämlich die Struktur der Konversation zwischen dem Anwender und dem Computersystem zu definieren.[56]

Bei der Gestaltung dieser Mensch-Rechner-Schnittstellen wurde schon früh erkannt, dass es ohne eine Normierung einen unübersehbaren Wildwuchs von Anwendungsoberflächen mit mehr oder weniger benutzer- und aufgabenorientierten Lösungen geben wird. Daher wurden von VDI (Verein Deutscher Ingenieure), DIN (Deutsches Institut für Normung) und diversen anderen Organisationen Normen für die Entwicklung von Benutzungsschnittstellen sowie für die Software-Ergonomie festgelegt, die unter anderem fünf Grundsätze zur Gestaltung von Benutzungsschnittstellen enthalten:[57]

---

[55] Vgl. Steinmetz 1993, S. 414ff

[56] Vgl. Dix et al. 1993, S. 296

[57] Ziegler 1993,S. 9ff

- Die *Aufgabenangemessenheit* bedeutet, dass der Benutzer bei der Erledigung seiner Arbeitsaufgaben nicht unnötig durch Eigenschaften der Benutzungsschnittstelle belastet wird. Die Aufgabenangemessenheit des Dialogs steht in sehr engem Zusammenhang mit der Aufgabenangemessenheit des gesamten Systems. Dies gilt umso mehr, je dialogintensiver sich das System gestaltet. Wichtige Faktoren für die Aufgabenangemessenheit des Dialogs sind die Antwortgeschwindigkeit des Systems und die Bereitstellung der notwendigen Funktionalität.

- Mit *Selbstbeschreibungsfähigkeit* ist gemeint, dass eine Benutzungsschnittstelle in ihrer Bedienung weitestgehend verständlich ist oder der Benutzer auf Verlangen zu dem jeweiligen Dialog eine Erläuterung erhalten kann. Das System benötigt also zur Bedienung keine dritte Instanz wie etwa ein gesondertes Benutzerhandbuch oder eine gesonderte Benutzerschulung, sondern es bietet die Möglichkeit, während des Arbeitens mit dem System das System direkt kennen und nutzen zu lernen. Im Rahmen der Selbstbeschreibungsfähigkeit werden zwei Grade unterschieden: Beim ersten Grad muss der Benutzer nicht explizit eine Erläuterung vom System verlangen. Das System stellt sich direkt im operativen Dialog so dar, dass seine Bedeutung und Handhabung unmittelbar klar sind. Techniken zur Schaffung einer Selbstbeschreibungsfähigkeit ersten Grades sind:

    1. Durch einen Menüdialog bietet das System eine Auswahl von Alternativen, die allerdings klar und eindeutig formuliert sein müssen.

    2. Die direkte Manipulation zeichnet sich dadurch aus, dass die zu bearbeitenden Objekte direkt (mit der Maus) verändert werden können. Durch ein Popupmenü sind zusätzliche Operationen dokumentiert und sofort ansprechbar.

    3. WYSIWYG[58] bezeichnet die Eigenschaft eines Dokumentes, seine Eigenschaften direkt und unverschlüsselt darzustellen. Eine zusätzliche Dokumentation zur Entschlüsselung des Dokuments ist nicht nötig.

[58] WYSIWYG – What You See Is What You Get

4. Durch einen graphischen Dialog können Sachverhalte mit Hilfe der graphischen Darstellung einfacher veranschaulicht werden als durch eine ausführliche verbale Darstellung.
5. Visual Feedback beschreibt die Fähigkeit eines Systems, jede Änderung an einem Objekt dem Nutzer sofort optisch darzustellen.

Die Selbstbeschreibungsfähigkeit zweiten Grades bedeutet, dass das System in jeder Dialogsituation die Möglichkeit zur Ausgabe von Hilfsinformationen bietet.

- *Steuerbarkeit* beschreibt die Möglichkeit eines Dialoges, Einfluss auf die Geschwindigkeit des Ablaufes sowie auf die Auswahl und Reihenfolge von Arbeitsmitteln zu nehmen. Der Benutzer kann also nach verschiedenen Gesichtspunkten den Dialog auf seine Bedürfnisse und die Aufgabenstellung anpassen. So kann es z.B. dem Benutzer möglich gemacht werden, innerhalb eines Dialoges verschiedene, für seine Aufgabe nicht notwendige Eingabefelder auszublenden. Ebenso sollte es möglich sein bei mehreren, sich wiederholenden Eingaben in einem Dialog, Eingabefelder, die sich von einer Eingabe zur nächsten nicht ändern, automatisch zu überspringen. Ein weiteres Mittel, um eine bessere Steuerbarkeit zu erreichen, ist es, Dialogsequenzen aufzuzeichnen, und diese bei Bedarf automatisch ablaufen zu lassen. Durch die Anpassung von Menüleisten und Steuerelementen an die Bedürfnisse des Benutzers kann eine zusätzliche Steuerbarkeit geschaffen werden.
- Die *Erwartungskonformität* bezeichnet die Eigenschaft, dass sich der Dialog genauso verhält, wie der Benutzer es von ihm erwartet. Innerhalb eines Systems sollten sich die verschiedenen Dialoge einheitlich verhalten. Des Weiteren sollten Dialoge so reagieren, wie der Benutzer es aus der Betrachtung seiner Umwelt erwarten würde, wodurch sich der Aufwand für Schulungen, Übungen und zur Sammlung von Erfahrungen verringert.
- Mit *Fehlerrobustheit* wird die Fähigkeit des Systems umschrieben, auf eventuelle Fehleingaben des Benutzers stabil und mit minimalem Korrekturaufwand zu reagieren. Es gibt verschiedene Möglichkeiten, Fehleingaben des Benutzers zu verarbeiten. Die beste Alternative ist es, Fehler

schlichtweg zu vermeiden. Dies wird durch konsequente Anwendung der o.g. Grundsätze zur Gestaltung von Benutzungsschnittstellen weitgehend erreicht. Eine weitere Möglichkeit ist die Fehlervorbeugung. Dies geschieht einerseits durch die Führung des Benutzers durch das System mittels Menüdialog und andererseits durch die Verwendung von graphischen Dialogen, welche Übertragungsfehler begrenzen. Eine weitere wichtige Maßnahme der Fehlervorbeugung ist, in kritischen Dialogsituationen den Benutzer noch einmal um eine Bestätigung seiner Aktion zu bitten (Sicherheitsabfrage). Sollten dennoch Fehler auftreten, muss das System in der Lage sein, diese zu erkennen und dem Benutzer die Möglichkeit der Korrektur zu geben. Dies beginnt damit, dass der Fehler durch eine passende Fehlermeldung ausgegeben wird, welche dem Vorstellungsmodell und der Sprache des Benutzers entspricht. Des Weiteren sollten dem Benutzer entsprechende Hilfeinformationen, die den Grund des Auftretens des Fehlers und eine mögliche Behebung erläutern, angezeigt werden.[59]

3.5 Verschiedene Kommunikationsmöglichkeiten zwischen Java- und Nicht-Java-Applikationen

Wie schon in der Einleitung betont, unterliegen moderne Informationssysteme der Forderung an größtmögliche Flexibilität, um an geänderte Geschäftsprozesse schnell anpassbar zu sein. Mit herkömmlichen Anwendungssystemen ist dieses nicht mehr realisierbar. Die technologischen Antworten heißen Client/Server Architektur und Komponenten-Software. Diese entstehenden Client/Server Architekturen und Softwarekomponenten haben sich in den meisten Fällen über längere Zeit entwickelt und sind heterogener Art. Um solche Systeme zu realisieren, werden auch an Programmiersprachen und Kommunikationstechnologien höhere Anforderungen gestellt.

Auf dem Markt existieren heutzutage diverse Programmiersprachen, die auf unterschiedliche Zielsetzungen ausgerichtet sind. Jede dieser Sprachen implementiert eigene Konzepte im Rahmen der Datenhaltung und des Datenaustausches. Dieser

[59] Vgl. Zeidler / Zellner 1994, S. 151ff

Sachverhalt impliziert häufig Schnittstellen- und Kommunikationsprobleme zwischen in unterschiedlichen Programmiersprachen entwickelten Programmen. Zumeist bieten die unterschiedlichen Entwicklungsumgebungen jedoch standardisierte Schnittstellen zu Datenhaltungskomponenten. Diese Komponenten, wie zum Beispiel Datenbanken und temporäre Dateien, bilden dann die Kommunikationsplattform für die verschiedenen Applikationen. Diese einfachen Kommunikationsmöglichkeiten sowie spezielle, standardisierte Kommunikationskanäle zwischen Anwendungen werden hier dargestellt.

3.5.1 Kommunikation über Datenbanken

Die meisten der derzeit existierenden Programmiersprachen bieten die Möglichkeit, direkt oder indirekt über ein Kommunikationsmodul wie zum Beispiel ODBC[60] auf Datenbanken zuzugreifen. Insofern können Datenbanken als Kommunikations-plattformen zwischen Applikationen agieren. Die aktuell häufig eingesetzten Formen von Datenbanken, relationale Datenbanken und objektorientierte Datenbanken, sollen hier kurz vorgestellt werden.

3.5.1.1 Kommunikation über relationale Datenbanken

Das System der relationalen Datenbank ist aktuell eine stark verbreitete Technologie zur Haltung von Daten. Namenhafte Hersteller sind z.B. Oracle, Sybase oder IBM. Relationale Datenbanken bestehen aus einem System von Tabellen, welche die zu speichernden Daten enthalten.

Die Spalten der Tabellen werden bei der Erstellung festgelegt und definieren Art und Größe der in ihnen zu haltenden Daten. In den Zeilen können dann die einzelnen Datensätze verwaltet werden, die allerdings in ihrem Aufbau alle der Struktur der Spalten entsprechen müssen. Zu den einzelnen Tabellen können Indizes angelegt werden, die den Zugriff auf die enthaltenen Daten erheblich beschleunigen.

Auf relationale Datenbanken können mehrere Benutzer gleichzeitig zugreifen und deren Daten auch zeitgleich bearbeiten, sofern der Zugriff nicht auf den selben Datensatz erfolgt. Relationale Datenbanksysteme bieten im Rahmen des Sicherheitsaspekts eine

[60] ODBC – Open Database Connect

Rechteverwaltung für die einzelnen Benutzer an, um nur zugelassenen Benutzern den Zugriff auf Daten zu gewähren.

Zu den Qualitäten von relationalen Datenbanken zählt insbesondere die Zugriffsgeschwindigkeit bei einer großen Menge von Datensätzen. In diesem Bereich sind relationale Datenbanken gegenüber anderen Datenbanken führend. Durch die Möglichkeit des parallelen Zugriffs sind sie zum Einsatz als Kommunikationsplattform geeignet, und die Verwaltung der Zugriffsrechte bietet eine entsprechende Sicherheit.

Die Kommunikation zwischen einer Applikation und einer relationalen Datenbank kann lokal auf einem System, aber auch über LAN- oder WAN-Netzwerke erfolgen. Der Informationsaustausch erfolgt mittels einer einheitlichen Datenbankabfragesprache, in der Regel SQL[61]. Mit Hilfe dieser Sprache kann die komplette Datenbank modifiziert werden. Häufig wird die Kommunikation über zwischengesetzte Module realisiert (ODBC), die anwendungsseitig eine standardisierte Schnittstelle bieten und datenbankseitig auf den entsprechenden Hersteller und Version der Datenbank abgestimmt sind. Programmiersprachen müssen also nur die anwendungsseitige Schnittstelle dieses Moduls implementieren und sind dann mit jeder Datenbank kommunikationsfähig.

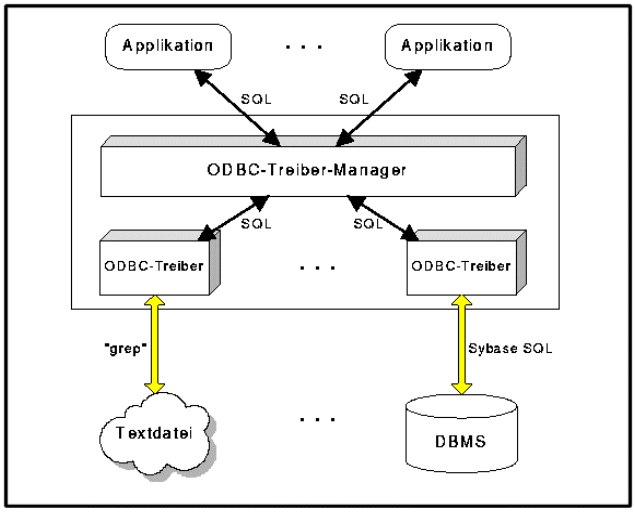

Abbildung 19: Kommunikation zwischen Applikationen und relationalen Datenbanksystemen

[61] SQL – Structured Query Language

Wenn als Programmiersprache Java eingesetzt wird, kann die Kommunikation nicht über ODBC alleine geschehen, sondern es muss das in Java integrierte JDBC[62] Modul eingesetzt werden. Dieses kann sowohl direkt mit einer Datenbank kommunizieren, als auch als Schnittstelle zwischen Java und ODBC fungieren. Eine weitere Anbindungsmöglichkeit an eine relationale Datenbank ist die über JDBC-Net, wobei die Datenbankanforderungen innerhalb der Applikation in ein datenbankunabhängiges Protokoll übersetzt werden. In diesem Fall muss dann datenbankseitig ein entsprechender Übersetzer angeboten werden. Die nachfolgende Grafik verdeutlicht diese Zusammenhänge.

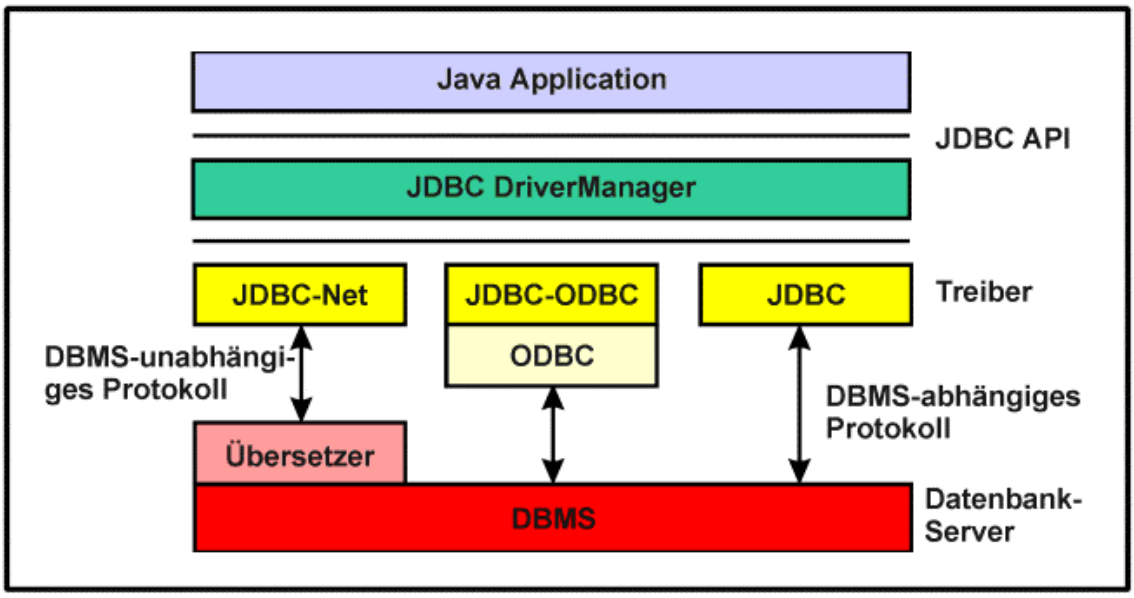

Abbildung 20: Kommunikation mit JDBC

Wenn zwei Applikationen miteinander über eine relationale Datenbank kommunizieren sollen, kann das realisiert werden, indem man eine Tabelle auf dem Datenbankserver anlegt, in der die auszutauschenden Informationen der Anwendungen wechselseitig abgelegt, ausgelesen und anschließend gelöscht werden. Ein Vorteil einer solchen Kommunikation liegt darin, dass die Implementation relativ einfach durchzuführen ist, da die Schnittstellen zu relationalen Datenbanken standardisiert sind. Ein weiterer

[62] JDBC - Java Database Connect

Vorzug ist es, dass im WAN Bereich eine solche Anwendung schnell Daten übermitteln kann, weil SQL-Befehle auf die im wesentlichen benötigten Daten beschränkt sind.

Als ein Nachteil ist zu nennen, dass die Applikationen in einem bestimmten Zeitintervall die Datenbanken auf neue Informationen abfragen müssen, da die Daten nicht direkt an die Applikationen weitergeleitet werden. Ein weiteres Manko ist die Einschränkung der Form der auszutauschenden Daten durch die bereits angelegte, vorstrukturierte Tabelle, über welche die Kommunikation realisiert wird.

3.5.1.2 Kommunikation über objektorientierte Datenbanken

Objektorientierte Datenbanken gewinnen im Zuge der Verbreitung von objektorientierten Programmiersprachen immer mehr an Bedeutung. Als einer der bekanntesten Hersteller von objektorientierten Datenbanken ist sicher die Firma Poet zu nennen.

Im Gegensatz zu den relationalen Datenbanken sind die objektorientierten dazu in der Lage, auch unstrukturierte Daten zu halten. Die von in objektorientierten Programmiersprachen erstellten Applikationen erzeugten Objekte (Vgl. auch Kapitel 3.3.3) können komplett an eine objektorientierte Datenbank übergeben werden, inklusive der im Objekt enthaltenen Daten und Methoden. Soll eine Kommunikation über eine objektorientierte Datenbank stattfinden, so wird im Wesentlichen wie bei den relationalen Datenbanken verfahren. Die an der Kommunikation beteiligten Applikationen greifen dann wechselseitig auf die objektorientierte Datenbank zu, wobei die Datenbank, wie auch schon bei den relationalen Datenbanken nicht lokal installiert sein muss, sondern auch im LAN oder WAN bereitstehen kann.

Vorteile dieser Kommunikationstechnik sind zum einen, dass objektorientierte Programmiersprachen wie z.B. Java unterstützt werden und zum anderen, dass beliebige Datenstrukturen variabel gespeichert werden können. Bei größeren Datenmengen werden die Abfragezeiten bei objektorientierten Datenbanken erheblich länger als bei den Relationalen. Außerdem ist auch bei diesem Kommunikationsmittel mit einzubeziehen, dass Applikationen in bestimmten Zeitscheiben die Datenbank auf neue Informationen prüfen müssten.

3.5.2 Kommunikation über dateibasierte Schnittstellen

Ein großer Nachteil der im obigen Kaptitel erwähnten Kommunikation über Datenbanken ist, dass ein entsprechendes, kostenintensives Datenbanksystem vorhanden sein muss. Eine kostengünstigere Alternative bietet die Kommunikation über ein Dateisystem. Wenn die Applikationen, die miteinander kommunizieren sollen auf einem System vorliegen, wird die Zugriffsteuerung vom Betriebssystem übernommen. Liegen die Anwendungen verteilt in einem Netzwerk vor, so muss diese Steuerung ein Netzwerkbetriebssystem wie z.B. NOVELL, MS-Windows NT oder auch UNIX / LINUX übernehmen.

Die Applikationen speichern die auszutauschenden Daten in einer Datei im Dateisystem ab, aus der sie von den Empfängern ausgelesen werden. Dieses System birgt allerdings im Rahmen der Zugriffssteuerung einige Probleme, wie z.B. die Verwaltung des Zugriffs bei mehreren Anforderungen durch Anwendungen.

Dieses System wird z.B. bei Paradox Datenbanken eingesetzt, welche ihre Daten auf Dateiebene ablegen. Jede Applikation, die einen Zugriff auf eine Paradox Datenbank vornehmen möchte, muss sich vorher in eine sogenannte Lock-Datei eintragen und die Tabelle und den gewünschten Datensatz in einer weiteren, für die Sperrung der Tabellen zuständigen Datei vermerken. Wenn eine weitere Anwendung auf diese Daten zugreifen möchte, so muss sie zuerst in dieser Lock-Datei überprüfen, ob der gewählte Datensatz zur Bearbeitung freigegeben ist. Wenn die Daten nicht gesperrt sind, so trägt sich diese Anwendung mit ihren Sperren zusätzlich zu den von anderen Applikationen vorgenommenen Sperren in die Lock-Datei ein.

Eine weitere Form des Datenaustausches bieten sogenannte Schnittstellendateien. Diese Dateien werden als dritte Instanz zwischen den Kommunikationspartnern genutzt. Eine solche Kommunikation wird vom Benutzer manuell durch den Export von Daten aus einer Applikation in eine Schnittstelle und den darauf folgenden Import der Daten in eine andere Applikation realisiert. Diese Kommunikation ist für vereinzelt auftretende Fälle geeignet, da meist ein größerer Aufwand für den Import der Daten notwendig ist.

3.5.3 Kommunikation über spezielle Verbindungstechnologien

Die Anwendung der oben genannten Kommunikationstheorien zieht einige Nachteile wie den Einsatz kostenintensiver Datenbankserver und den stark erhöhten

Netzwerkverkehr bei deren Nutzung in einer LAN- oder WAN-Umgebung nach sich. Sinnvoll ist eher eine direkte Kommunikation zwischen zwei in unterschiedlichen Programmiersprachen entwickelten Anwendungen anstatt des Informationsaustausches über dritte Kommunikationsplattformen. Auch in diesem Bereich existieren zwei bekannte Lösungen: CORBA[63] und COM[64], die hier im Folgenden vorgestellt werden.

3.5.3.1 Kommunikation über CORBA

CORBA, die Common Object Request Broker Architecture wurde von der OMG[65], einem Standardisierungsgremium mit mehr als 700 Mitgliedern, 1991 in der ersten Version definiert. CORBA war eine Antwort auf die starke Zunahme von Hardware- und Software-Produkten, mit dem Ziel, ein Kommunikationsmedium zu schaffen, welches eine orts-, plattform- und implementationsunabhängige Kommunikation zwischen Applikationen erlaubt.

Die technische Implementation des Standards CORBA ist der sogenannte ORB[66]. Ein ORB ermöglicht es einem Client eine Meldung transparent an ein Serverobjekt zu senden, wobei das Serverobjekt auf derselben oder einer anderen Maschine vorhanden sein kann. Der ORB ist dafür zuständig, das Serverobjekt zu finden, dort die entsprechende Funktion aufzurufen, die Parameter zu übergeben und das Resultat an den Client zurückzureichen. Dadurch wird eine nahtlose Interoperabilität zwischen Applikationen erreicht, welche in einem völlig heterogenen Umfeld betrieben werden können.

Bisher wurde in einem heterogenen Umfeld typischerweise jede Schnittstelle spezifisch programmiert. Dabei mussten Plattform, Betriebssystem, Programmiersprache und anderes in Betracht gezogen werden. Bei Änderungen ist die Anpassung solcher Schnittstellen entsprechend schwerfällig. Bei CORBA besteht eine strikte Trennung zwischen der Schnittstellendefintion eines Objektes und deren Implementation. Beim CORBA-Vorgehen wird zunächst die öffentliche Schnittstelle eines Objektes (d.h. die

[63] CORBA – Common Object Request Broker Architecture

[64] COM – Component Object Model

[65] OMG – Object Management Group

[66] ORB – Object Request Broker

Funktionen) in der Interface Definition Language (IDL) definiert, die eine implementationsunabhängige Beschreibungssprache darstellt. In einem zweiten Schritt wird diese Definition ausprogrammiert und zwar sowohl für den Client-, als auch für den Server-Teil. Dabei kann der Client in einer anderen Programmiersprache implementiert werden als der Server.[67]

Die Kommunikation zwischen den Applikationen und dem ORB übernehmen zwei zwischengeschobene Objekte. Auf der Seite des Clients ist das der sogenannte Client-Stub, serverseitig der Server-Skeleton. Die Client Applikation ruft eine Objektreferenz auf, die vom Client-Stub abgefangen wird und an den ORB übergeben wird. Der ORB lokalisiert das angesprochene Objekt mit Hilfe eines zu CORBA gehörigen Namensdienstes und leitet den Methodenaufruf an den auf dem entsprechenden System laufenden Server-Skeleton weiter. Dieser wiederum greift auf das in seinem System existierende Objekt zu und übergibt die Rückgabeparameter zurück an den ORB, welcher diese Informationen wiederum an den Client zurückliefert. Die folgende Abbildung zeigt diesen Ablauf.

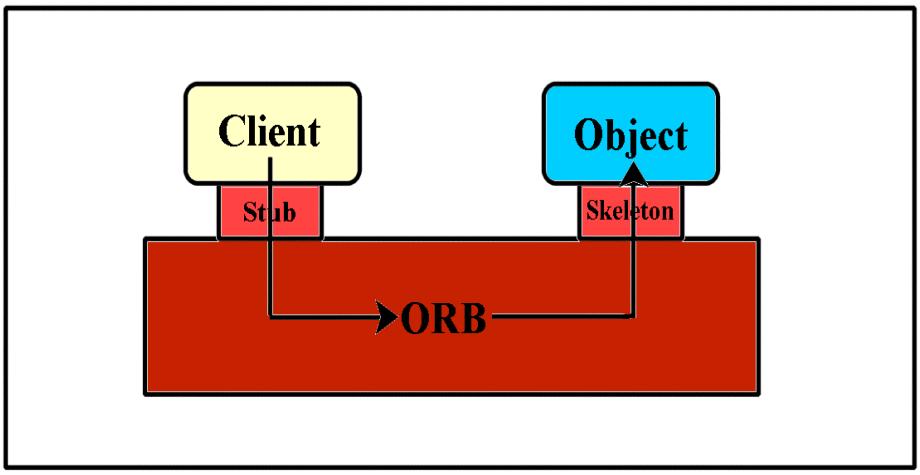

Abbildung 21: Die Kommunikation über den ORB[68]

Über den ORB ist auch eine Kommunikation in verteilten Systemen, z.B. über das Internet möglich. In diesem Fall kommt das Internet Inter ORB Protocol (IIOP) zum Einsatz, welches die innerhalb eines ORBs entstehenden Objektanfragen, welche nicht

[67] Vgl. OMG

[68] Vgl. OMG 2000

innerhalb dieses ORBs aufgelöst werden können, an den entsprechenden ORB weiterleitet, in dem die Objekte vorhanden sind.

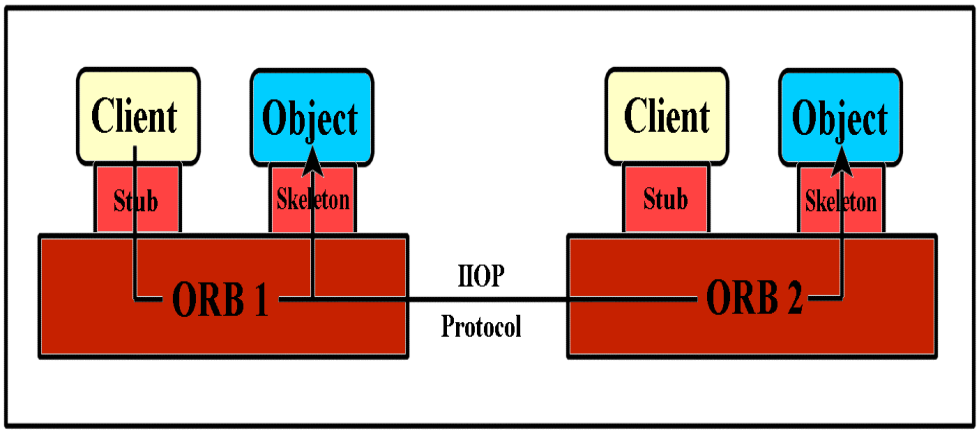

Abbildung 22: Kommunikation zwischen zwei ORBs mit IIOP[69]

3.5.3.2 Kommunikation über COM

Eine ähnliche Zielsetzung wie CORBA verfolgt auch Microsofts COM-Technologie, die in Zusammenarbeit mit der Digital Equipment Corp. entwickelt wurde, um eine Infrastruktur für objektorientierte Kommunikation innerhalb der Windows-Welt zu schaffen.

Das Objektmodell von Microsoft baut auf OLE[70] auf. COM stellt einen sprachunabhängigen, binären Standard für die Implementierung von Objekten zur Verfügung. Dieser Standard basiert auf der Auffassung, dass ein Objekt eine Menge von Funktionen implementiert, deren Untermengen eine zusammenhängende Sicht auf das Objekt ermöglichen. Diese Untermengen werden Interfaces genannt, wobei ein Objekt ein oder mehrere dieser Interfaces implementieren kann.

Die Realisierung eines Interfaces ist ein Feld von Zeigern auf die Funktionen der Objekte. Benutzer erreichen ein COM-Objekt immer durch ein solches Interface und nicht durch die Zeiger auf das vollständige Objekt. Diese Struktur beschreibt das Kernstück des binären Standards von COM.

---

[69] Vgl. OMG 2000

[70] OLE – Object Linking and Embedding

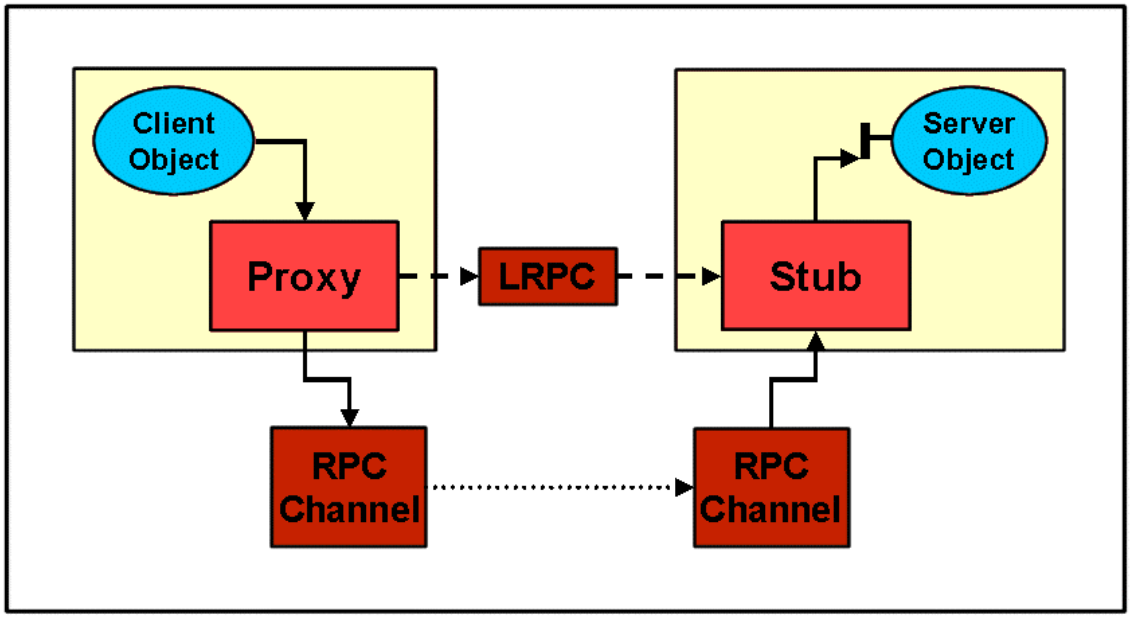

Abbildung 23: COM-Interfaces

Die obige Abbildung demonstriert die verschiedenen Abläufe des Methodenaufrufes im Component Object Model. Die Schnittstellen zum Übertragungsprotokoll werden ähnlich wie bei CORBA realisiert. Im Gegensatz zu CORBA werden hier die Schnittstellen clientseitig als *Proxy* und serverseitig als *Stub* bezeichnet[71]. Wird ein Methodenaufruf über Systemgrenzen hinweg angestrebt, kommt das RPC[72] Protokoll zum Einsatz, welches im Netzwerk einsetzbar ist. In diesem Fall wird das gesamte System auch mit dem Namen DCOM[73] betitelt. Falls sich die kommunizierenden Prozesse auf demselben System befinden, wird zur Performanceverbesserung das von Microsoft bereitgestellte LRPC[74] Protokoll verwendet. Dieses Protokoll produziert im Gegensatz zum RPC Protokoll einen geringeren Overhead, da keine netzwerkrelevanten Daten transportiert werden müssen.

[71] bei CORBA : “Client-Stub” und “Server-Skeleton”

[72] RPC – Remote Procedure Call

[73] DCOM – Distributed Component Object Model

[74] LRPC – Lightweight Remote Procedure Call

Dieses Konzept macht prinzipiell den Einsatz einer separaten Interface Definition Language notwendig. Da COM eine Microsoftentwicklung ist, ist die verwendete IDL auch eine Microsofteigene[75] und nicht zu anderen IDLs kompatibel.

3.5.3.3 Vergleich von CORBA und COM

COM unterstützt nur einen binären Standard für die Interoperabilität, der das Format des Nachrichtenverkehrs zwischen den Prozessen festlegt. Microsoft liefert zwar einen Precompiler, der die Generierung der Stubs für die Aufrufe bei sich kreuzenden Prozessen erleichtert - dieser ist aber eben nicht sprachunabhängig. CORBA dagegen unterstützt neben einem binären Standard auch andere Standards. Dieses ist schon wegen des hohen Anspruchs der Portierung notwendig. Durch den Einsatz einer höheren Sprache ist die dem Objekt zugrundeliegende Middleware abgeschirmt. Das bedeutet, dass durch CORBA der Grad der Abstraktion von einem binären Standard (mit Zeigern auf Tabellen im Speicher) hin zu einem Konstrukt einer höheren Sprache verschoben wird. Dadurch kann der Entwickler in seiner Sprache mit entfernten Objekten umgehen. Die Interoperabilität geht so weit, dass eine CORBA-Komponente, die in einer bestimmten Programmiersprache realisiert wurde, eine andere Komponente aufrufen kann, die in einer anderen Programmiersprache geschrieben wurde und von einem *fremden* ORB stammen kann. Keine andere Client/Server Middleware hat einen vergleichbaren Grad an Abstraktion und Möglichkeiten der Portierung erreicht. Im Gegensatz dazu ist COM nur auf die Microsoft-Welt ausgerichtet.[76]

3.5.3.4 Extensible Markup Language (XML)

XML wurde im Februar 1998 vom World Wide Web Consortium (W3C) als Standard verabschiedet und besteht aus einem Teil des bereits seit zehn Jahren existierenden Sprachkonzepts SGML[77], aus dem auch das zum Datenaustausch im Internet bewährte Kommunikationskonzept HTML[78] entstand.

---

[75] Die von Microsoft endwickelte IDL: MIDL - Microsoft Interface Definition Language

[76] Vgl. Bark 1996.

[77] SGML – Standard Generalized Markup Language

[78] HTML – Hypertext Markup Language

Die Extensible Markup Language ist eine textbasierte Meta-Auszeichnungssprache, die es ermöglicht, Daten bzw. Dokumente derart zu beschreiben und zu strukturieren, dass sie, insbesondere über das Internet, zwischen einer Vielzahl von Anwendungen ausgetauscht und weiterverarbeitet werden können.

Mit XML können die Struktur und der Inhalt von Dokumenten so präzise beschrieben werden, dass es letztlich nicht mehr notwendig ist, die zum Verständnis und der Weiterverarbeitung von Daten notwendigen Informationen fest in die Anwendungen zu integrieren. Vielmehr besteht die Vision darin, den auszutauschenden Daten die zu ihrer Nutzung notwendigen Informationen mitzugeben. Neben der Unterstützung des Dokumentenaustausches ermöglicht XML allgemein die flexible Wiederverwendung von Daten und ist die Grundlage für eine Verwendung von Metadaten, was unter anderem zu einer erhöhten Interoperabilität unterschiedlicher Anwendungen führt.[79]

Das XML-Format ist dem von HTML ähnlich. Im Kopf des XML-Dokumentes werden grundlegende Attribute festgelegt, wie z.B. der verwendete Textcodierungsstandard. Danach folgt ein sogenanntes Wurzelelement, an dem sämtliche weiteren Elemente hängen. Dieses wird am Ende des Dokumentes durch einen Slash (/) geschlossen. Innerhalb des Wurzelelementes werden dann praktisch die einzelnen Datensätze aufgeführt, die auch jeweils mit einem Slash beendet werden müssen. Der einzelne Datensatz besteht dann wiederum aus den einzelnen Feldern, die ihn bestimmen. Ein Beispiel gibt die folgende Abbildung.

[79] Vgl. Weitzel / Franke / Hittmeyer

```
<buch>
 <titel>XML</titel>
 <ueberblick>
  <titel>Struktur XML</titel>
  <absatz>
   Vorwort.
  </absatz>
 </ueberblick>
 <kapitel>
  <kapitel1>
   <titel>Was ist XML</titel>
   <absatz>
      XML ist eine Technologie zum....
   </absatz>
  </kapitel1>
 </kapitel>
</buch>
```

Abbildung 24: Ein XML Beispiel

Solche XML Daten werden von einem sogenannten „Parser“, einem Teilmodul der Anwendung, zerlegt und dem Anwendungsprogramm zur weiteren Verarbeitung zur Verfügung gestellt.

XML stellt also einen Kommunikationsstandard dar, mit dem Applikationen Daten austauschen können. Diese XML-Daten können sowohl in Dateiform bereitstehen, als auch über einen Datenstrom, wie ihn z.B. das Internet bietet, ausgetauscht werden.

Auch das für diese Arbeit grundlegende GroupProcess-Projekt setzt bereits XML zur Speicherung der Workflow-Daten ein. Dadurch wird eine spätere Nutzung der Workflow-Daten durch eine Drittapplikation ermöglicht.

3.5.4 Spezielle Kommunikationsmöglichkeiten von Lotus Notes

Lotus Notes bietet ein vielfältiges Spektrum von Kommunikationsmöglichkeiten an. Die in den vorherigen Teilabschnitten des Kapitels 3.5 bereits dargestellten Kommunikationsmöglichkeiten zwischen Nicht-Java-Applikationen werden von Lotus Notes teilweise unterstützt, wobei eine Kommunikation durch CORBA insbesondere

durch die Einbindung der Java-Technologie in Lotus Notes ermöglicht wird. Die relationalen Datenbankkonzepte werden durch sogenannte @-Funktionen unterstützt. @-Funktionen sind von Lotus Notes bereitgestellte Funktionen, die eine einfache Programmierung von Lotus Notes ermöglichen. Diese @-Funktionen greifen wiederum über den entsprechenden ODBC-Treiber[80] auf die jeweilige relationale Datenbank zu.

Der Nachrichtenaustausch über temporäre Dateien sowie über COM wird in der, im Gegensatz zu den @-Funktionen, eher etwas komplizierteren Lotus Script Sprache realisiert.

Zusätzlich zu diesen Kommunikationsmöglichkeiten bietet Lotus Notes ab der Version 5 durch die Bereitstellung eigener Java-Klassen zum Zugriff auf Notes-Datenbanken, Java-Applikationen die Möglichkeit direkt auf Datenbanken zuzugreifen. Diese Java-Klassen sind in einem Paket zusammengefasst, dass von den entsprechenden Applikationen importiert werden kann. Dadurch können auch Java-Applikationen implementiert werden, die unabhängig von Lotus Notes laufen, also nicht als Applet in ein Lotus Notes-Dokument integriert sein müssen. Damit sind die Applikation in der Lage, auf sämtliche Lotus Notes-Datenbanken zuzugreifen sowohl auf solche die auf Lotus Domino-Servern abgelegt sind als auch auf lokal gespeicherte. Allerdings wird für den entfernten Zugriff auf Datenbanken wiederum CORBA über IIOP als Kommunikationsweg eingesetzt, wobei allerdings in diesem Fall die Implementierung des CORBA Interfaces durch den Programmierer entfällt, da diese bereits in den von Lotus Notes bereitgestellten Klassen erfolgt ist.

Durch die Verwendung dieser Java-Klassen lässt sich die Performance von Applikationen, die auf Lotus Notes-Datenbanken zugreifen erheblich verbessern, da kein Umweg über ein drittes Kommunikationsmedium genommen werden muss (mit Ausnahme der entfernten Kommunikation, wegen des Einsatzes von CORBA). Genauso verringert sich der Implementierungsaufwand, da der Programmierer sich nicht, wie z.B. bei einer Lösung über relationale Datenbanken, mit der Implementierung einer weiteren Programmiersprache beschäftigen muss, wie z.B. SQL.

[80] Siehe auch Kapitel 3.5.1

Allerdings wird das Einsatzgebiet einer Anwendung, die diese Java-Klassen verwendet kleiner, da diese ausschließlich mit Lotus Notes-Datenbanken kommunizieren können. Eine solche Applikation sollte also nur zur Benutzung mit Lotus Notes vorgesehen werden, oder ein einfach zu ersetzendes Kommunikationsmodul einsetzen, welches bei einem Einsatz in anderen Umgebungen ausgetauscht wird.

3.6 Kommunikation zwischen Java-Applikationen

Die im vorherigen Kapitel erläuterten Kommunikationsmöglichkeiten sind applikations- und sprachenunabhängig einsetzbar. Des Weiteren existieren im Bereich der Java-Technologie Kommunikationstechniken, die speziell auf die Kommunikation zwischen Java-Applikationen zielen. Beim systemübergreifenden Informationsaustausch zwischen Applikationen kommt RMI zum Einsatz, beim systeminternen kann die Interface-Technologie genutzt werden. Diese beiden Techniken werden hier vorgestellt.

3.6.1 Kommunikation über RMI

Bei der Kommunikation zwischen verschiedenen Java-Applikationen tritt das Problem auf, wie der Zugriff auf Objekte der jeweils anderen Applikation realisiert werden soll. In diesem Szenario fällt der Kommunikation in verteilten Systemen eine besondere Rolle zu, da sie die Realisation von verteilten Anwendungen, zum Beispiel im Intra- oder Internet, ermöglicht.

Bei Java stehen für diese Art des Datenaustausches spezielle Klassen zur Verfügung: die RMI-Klassen. Hierbei handelt es sich um ein Objektmodell und eine Menge von APIs, die zur Realisierung von solchen verteilten Systemen in Java bereitstehen.

Verteilte Systeme im objektorientierten Umfeld erfordern jedoch ein verteiltes Objektmodell und einen verteilten Nachrichtenfluss, welcher einem entfernten Methodenaufruf entspricht. Der entfernte Methodenaufruf, welcher in der Welt der prozeduralen Programmiersprachen auch unter RPC bekannt ist, ist bei objektorientierten Sprachen, wie auch Java eine ist, nicht oder nur eingeschränkt verwendbar. Mit RMI steht Java eine Transferierung der RPC-Konzepte zur Verfügung, welche die besonderen Belange der objektorientierten Sicht berücksichtigt.

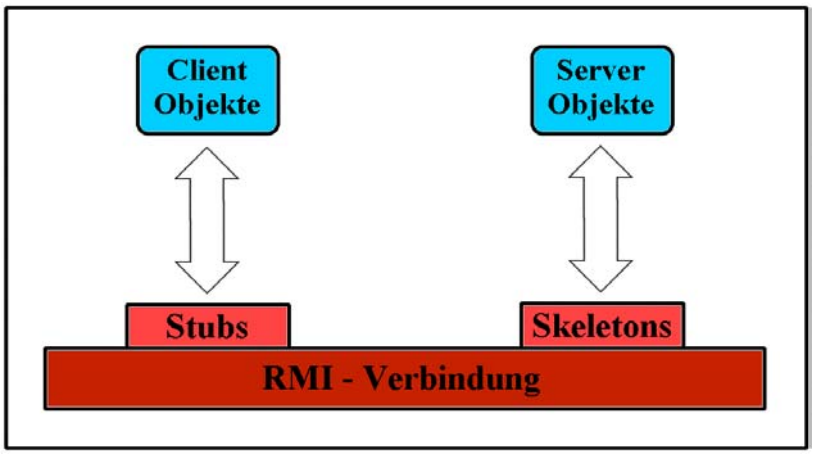

Abbildung 25: Kommunikation über RMI

Dabei wird die Kommunikation nicht direkt auf die Objekte ausgeführt, sondern über sogenannte Stellvertreterobjekte. Das Stellvertreterobjekt auf Seiten der Client-Applikation wird wie auch bei CORBA „Stub“ genannt und existiert im Adressraum der Java Virtual Machine, so dass es für die restliche Applikation ein reguläres Java-Objekt darstellt. Das entsprechende Stellvertreterobjekt auf Seiten des Servers wird als „Skeleton“ bezeichnet. Auch dieses Objekt existiert, wie auch der Stub, in der lokalen Virtual Machine und kann deshalb direkt von der restlichen Applikation wie ein orginäres Objekt behandelt werden. In den Stubs und Skeletons wird festgelegt, auf welche Objekte und Methoden remote Zugriff gewährt wird. Diese Eigenschaften machen aus den Stellvertreterobjekten Schnittstellen für objektorientierte und verteilte Kommunikation. Dabei ist es nicht notwendig, dass die kommunizierenden Virtual Machines auf verschiedenen Systemen liegen.[81]

3.6.2 Kommunikation über Interfaces

Interfaces[82] spielen bei der Programmierung in Java eine wichtige Rolle, denn bei Java ist im Gegensatz zu anderen objektorientierten Programmiersprachen eine Mehrfachvererbung nicht möglich. Hier werden als Ersatz Interfaces angeboten, deren Technik im Nachfolgenden erläutert wird.

[81] Vgl. Schmidt 1997

[82] Interface: übers. Schnittstelle

Schnittstellen sind in Java abstrakte Klassen, was bedeutet, dass sie keinen Quellcode innerhalb der Methoden enthalten, sondern nur das Gerüst für diese festlegen. Verschiedene Klassen können dann ein oder mehrere Interfaces nutzen. Die Nutzung eines Interfaces bedeutet, dass alle Methoden, die in der Schnittstelle definiert wurden, in der betreffenden Klasse implementiert werden müssen. Objekte werden also typisiert, indem sie ein Interface implementieren.

„Java-Schnittstellen können gemäß dem OOP-Konzept der Vererbung auch andere Schnittstellen erweitern, um somit zuvor geschriebene Schnittstellen weiterentwickeln zu können. Die neue Subschnittstelle erbt dabei auf die gleiche Art und Weise wie bei Klassen die Eigenschaften von der/den Superschnittstelle(n). Vererbt werden alle Methoden und statischen Konstanten der Superschnittstelle.

Zwar können Schnittstellen andere Schnittstellen erweitern, aber weil diese keine konkreten Methoden definieren dürfen, dürfen auch die Subschnittstellen keine Methoden der Superschnittstellen definieren. Statt dessen ist dieses die Aufgabe jeder Klasse, welche die abgeleitete Schnittstelle verwendet. Die Klasse muss sowohl die in der Subschnittstelle deklarierten Methoden als auch alle Methoden der Superschnittstelle definieren.“[83]

Wenn z.B. eine Klasse ein bestimmtes Interface implementiert, in dem Methoden abstrakt definiert sind, so wird für Algorithmen, die mit einem Objekt dieser Klasse arbeiten garantiert, dass die in dem Interface definierten Methoden innerhalb dieses Objektes zur Verfügung stehen. Dadurch können unterschiedliche Klassen, die das gleiche Interface implementieren nach außen als die gleiche Klasse angesehen werden. Dies gilt allerdings nur im Bezug auf die im Interface definierten Methoden.

Um Interfaces für die flexible Kommunikation zwischen Java-Komponenten nutzen zu können, müssen die im Interface definierten Methoden zur Datenübergabe vorgesehen sein und das übergebende Objekt muss einen Zeiger auf das datenempfangende Objekt erhalten, welcher vom Typ des Interfaces ist. Die Klasse des empfangenden Objekts muss dann wiederum dieses Interface implementieren.

---

[83] Steyer 1998, S. 379

Diese Technik kann nur innerhalb eines lokalen Systems eingesetzt werden, da die Zeiger auf die Objekte im Hauptspeicher gehalten werden und nicht über ein Netzwerk transportiert werden können.

4 Anforderungen, Merkmale und Lösungsansatz eines OrganizationchartViewers und eines WorkgroupViewers für ein Ad-hoc-Workflow-Modellierungswerkzeug

Grundlage der in diesem Kapitel betrachteten Werkzeuge zur Betrachtung und Auswahl von Abteilungen und Arbeitsgruppen bildet das in Kapitel 2.4 beschriebene GroupProcess-Projekt. Darauf aufbauend beschäftigt sich dieses Kapitel mit einer Erweiterung dieses Projektes um eine Möglichkeit der Darstellung von Organisationsstrukturen innerhalb des in GroupProcess entwickelten Ad-hoc-Workflow-Modellierungswerkzeuges. Das GroupProcess-Projekt wird bei unterschiedlichen Typen von Benutzern eingesetzt, wodurch sich besondere Anforderungen an das Design und die Bedienung der neu zu entwickelnden Darstellungskomponenten ergeben.

Eine zusätzliche Herausforderung ergibt sich durch die Notwendigkeit der Integration diverser Auswahlmöglichkeiten in den OrganizationchartViewer und in den WorkgroupViewer. Diese sollen sowohl komplette Abteilungen und Arbeitsgruppen, als auch einzelne Personen dieser Organisationseinheiten umfassen und müssen, um dem Anspruch des Einsatzes bei unterschiedlichen Benutzertypen gerecht zu werden, möglichst einfach in ihrer Bedienung realisiert werden. Ein Überblick über eine generelle Einteilung von Benutzertypen wird am Anfang dieses Kapitels gezeigt.

Aufbauend auf dieser Einteilung und den grundsätzlichen in Kapitel 3.4 vorgestellten Anforderungen an Benutzungsschnittstellen und Dialoge ergeben sich die Gesamtanforderungen an die beiden neuen Anzeige- und Selektionsoberflächen. Diese werden im Unterkapitel 4.2 festgelegt.

Um die in der Organisationsdatenbank von PAVONE vorliegenden Strukturen der Organisation im OrganizationchartViewer bzw. WorkgroupViewer anzeigen zu können, muss ein Kommunikationsmechanismus geschaffen werden, der eine Übertragung

dieser Daten in das GroupProcess-Projekt gewährleistet. Ähnlich verhält es sich mit den von den Benutzern selektierten Organisationseinheiten, welche nach der Bestätigung der Auswahl an den Ad-hoc-WorkflowModeler des GroupProcessProjektes zur Weiterverwendung übergeben werden müssen. Basierend auf den in den Kapiteln 3.5 und 3.6 aufgezeigten möglichen Verbindungstechnologien zwischen Applikationen wird im Abschnitt 4.3 eine Lösung für diese Problematik erarbeitet.

4.1 Kontinuum der Nutzertypen von Organisationsdesignwerkzeugen

Im Rahmen des Kontinuums der Nutzertypen werden Mitarbeiter einer Organisation nach ihren unterschiedlichen Anforderungen an Organisationsmodellierungswerkzeuge unterschieden. Eine solche Unterscheidung wurde schon in [Ott/Nastansky 1998] gegeben und von Dipl.-Inform. Carsten Huth in dessen Diplomarbeit 1998[84] verfeinert. Diese Unterscheidung erfolgt in fünf Klassen, die hier im einzelnen kurz erwähnt werden sollen. Die Einordnung der Klassen erfolgt nach je nach Zugriffsart und -häufigkeit.

Die Anwenderklasse *Intensive Änderungen* bildet die Gruppe der klassischen Organisatoren einer Unternehmung ab. Typische Aufgaben solcher Organisatoren sind z.B. die Änderung der grundlegenden Organisationsstruktur, z.B. bei einer Änderung der Kerngeschäftsfelder einer Unternehmung. Diese Anwender nutzen ein entsprechendes Werkzeug auf der Strukturebene. Sie beschäftigen sich nicht mit der Zuweisung von Rollen oder z.B. der Einteilung von Personen in Abteilungen oder Arbeitsgruppen. Ein solcher Organisator nutzt demnach ein Werkzeug intensiv und stellt auch dementsprechend hohe Anforderungen an ein solches.

In der Anwenderklasse *Häufige Änderungen* werden Personen zusammengefasst, die hauptsächlich in der mittleren Managementebene zu finden sind, also im taktischen Management tätig sind. Diese Gruppe beschäftigt sich z.B. mit Teilbäumen innerhalb einer Organisationsaufbaustruktur. Es können z.B. einzelne Abteilungen aufgelöst, neu erstellt oder erweitert werden. Solche taktischen Änderungen der Organisationsstruktur

[84] Huth 1998

finden im allgemeinen häufiger statt als Änderungen innerhalb der strategischen Planung. Ebenso steigt auch der Detaillierungs- und Differenzierungsgrad.

Im Rahmen der Gruppe der *sporadischen Anpassungen* werden Mitarbeiter eines Unternehmens betrachtet, die zur unteren Führungsebene gehören und sich daher mit dem operativen Management befassen. An der Organisationsaufbaustruktur werden hier abteilungsübergreifend keine weiteren Änderungen mehr vorgenommen, sondern eher abteilungsintern. Die Tätigkeit ist in geringerem Maße planend als in den höheren Ebenen. Es wird mehr sporadisch auf aktuelle Gegebenheiten eingegangen. Im Rahmen der Abteilungsverwaltung ist hier davon auszugehen, dass die Nutzungshäufigkeit dieser Gruppe im Verhältnis zu den höheren Ebenen zurückgeht. Bei starker Arbeitsgruppenbildung innerhalb der Organisation ist mit einer zunehmenden Nutzung der Werkzeuge zu rechnen.

Mit *Pflege der eigenen Daten* wird die Gruppe tituliert, die Mitarbeiter enthält, die keinen eigenen Managementauftrag haben. Für eine Organisation ist es sinnvoll, Werkzeuge vorzuhalten, mit denen Mitarbeiter ihre persönlichen Daten eigenverantwortlich pflegen können, um den Verwaltungsaufwand zu reduzieren. Diese Personen nutzen ein entsprechendes Werkzeug zur Pflege der persönlichen Daten wie Adresse, Telefonnummer, E-Mail-Adresse usw. Da sich die zu bearbeitenden Daten in der Regel nicht sehr häufig ändern, ist mit einer geringeren Nutzungshäufigkeit eines solchen Gestaltungswerkzeuges zu rechnen. Ebenso ist auch die Nutzungsintensität gering, da weder einheitsintern, noch einheitsübergreifend Änderungen vorgenommen werden.

In der Gruppe *Informierender Zugriff* werden Mitarbeiter zusammengefasst, die rein lesenden Zugriff auf die in den oberen Klassen bereitgestellten Daten benötigen. Durch den Umstand, dass diese Gruppe von Nutzern nur lesend auf die Organisations-strukturdaten zugreift, kann von einer Nutzungsintensität von null ausgegangen werden, da unter dem Begriff „Intensität“ nur modifizierende Zugriffe verstanden werden. Bei der Nutzungsfrequenz ist festzustellen, dass sie in allen Managementebenen konstant ist. Zusätzlich ist zu erwähnen, dass Nutzer im Bereich des lesenden Zugriffs nicht nur organisationsintern existieren müssen. Ebenso gibt es die Möglichkeit, z.B. im Rahmen

eines virtuellen Unternehmens, dass auch organisationsexternen Benutzern der lesende Zugriff auf die Organisationsstrukturdaten genehmigt wird.[85]

4.2 Benutzerinterface für einen Abteilungsviewer und einen Arbeitsgruppenviewer

Durch die Integration der beiden Benutzeroberflächen des Organization- / WorkgroupViewers in das bestehende GroupProcess-Projekt, welches auf die Nutzung durch unterschiedliche Benutzertypen innerhalb einer Organisation abzielt, ergibt sich, dass die Handhabung der neu zu entwickelnden Werkzeuge möglichst einfach zu gestalten ist, um auch Nutzern mit geringen Kenntnissen die Anwendung zu ermöglichen. Darüber hinaus sollen hier die Grundsätze und Techniken der Gestaltung von Benutzungsschnittstellen und Dialogen mit einfließen.

Daraus entstehen die folgenden, für die Erstellung der Benutzeroberflächen relevanten Punkte:

1. Der Aufruf der beiden Werkzeuge soll aus dem Ad-hoc-Workflow-Modellierungswerkzeug des GroupProcess-Projektes erfolgen. In der Kombination mit den oben erläuterten Grundsätzen der Benutzungsschnittstellenmodellierung ist also die Handhabung der beiden zu erstellenden Werkzeuge an die bereits existierende Modellierungsumgebung anzupassen.
2. Für die Struktur des OrganizationchartViewers soll ein einfaches, hierarchisches Modell, ähnlich dem, welches von PAVONE im OrganizationModeler realisiert wurde, verwendet werden. Einerseits wird dadurch sichergestellt, dass die Umgebung - in diesem Fall die Organization Database - weiter genutzt werden kann und anderseits findet diese klassische Hierarchie (Einliniensystem) in einer Vielzahl der heutigen Organisationen ihre Anwendung. Die Oberfläche des OrganizationchartViewers zur Darstellung dieser Hierarchie soll weitestgehend vom PAVONE OrganizationModeler übernommen werden, weil dieser bereits teilweise unter den Mitarbeitern innerhalb einer Organisation bekannt ist und

[85] Vgl. Huth 1998, S. 39ff

das Erscheinungsbild gut in einen Kontext mit dem GroupProcess-Projekt zu bringen ist. In der Realisation von PAVONE ist momentan die Liste der Mitglieder und anderer Optionen einer Abteilung recht umständlich über ein sich neu öffnendes Fenster mit mehreren Registern zu erreichen. In dem zu entwickelnden OganizationchartViewer soll dies zur besseren Benutzerführung durch einen einfachen Mausklick auf die entsprechende Abteilung gelöst werden, durch den im unteren Bereich der gewählten Abteilung eine Liste der enthaltenen Personen angezeigt wird.

3. Das Arbeitgruppen-Modell im OrganizationModeler von PAVONE unterscheidet sich in seinem optischen Erscheinungsbild sehr stark von dem der Abteilungshierarchie. Aufgrund der in Kapitel 3.4.2 genannten Forderungen an die Dialoggestaltung sollten sich Benutzerschnittstellen innerhalb eines Systems einheitlich präsentieren. Insofern ist eine Gestaltung zu wählen, die jener der Abteilungshierarchie nahe kommt und gleichzeitig den Anforderungen an Arbeitsgruppen nachkommt. Die Arbeitsgruppen sind aber nicht fest in einer Hierarchie anzuordnen, sondern frei und ohne Verbindungen zueinander. Außerdem soll die Darstellung der Arbeitsgruppen so wie auch bei den Abteilungen durch Rechtecke erfolgen, wobei allerdings eine Unterscheidung zu den Abteilungen durch abgerundete Ecken möglich gemacht werden soll.

4. Durch den Einsatz einer Statusleiste am unteren Rand des Fensters soll dem Anspruch an die in Kapitel 3.4.2 erwähnte Selbstbeschreibungsfähigkeit zweiten Grades genüge getan werden. Hier sollen dem Benutzer zusätzliche Informationen zur Bedienung der Applikation angeboten werden.

5. In beiden Werkzeugen soll die Möglichkeit geschaffen werden sowohl einzelne Personen aus den entsprechenden Abteilungen bzw. Arbeitgruppen sowie auch die gesamten Organisationseinheiten auszuwählen. Diese Auswahlmechanismen sollen allesamt mit Hilfe der Maussteuerung realisiert werden. Im Falle einer Fehlauswahl muss es dem Benutzer ermöglicht werden, diese rückgängig zu machen.

6. Aufgrund der geforderten Integration in das GroupProcess-Projekt ist es notwendig, dass auch diese Werkzeuge plattformunabhängig einsetzbar und mit

Groupware kombinierbar sind. Aus diesem Grund und wegen der notwendigen Kommunikation zwischen den einzelnen Modulen (siehe Kapitel 4.3) sollte die Implementation dieser Werkzeuge mit Hilfe der Java-Technologie vorgenommen werden.

4.3 Kommunikation des OrganizationchartViewers / WorkgroupViewers mit dem GroupProcess-Projekt und Lotus Notes

Um den Einsatz der beiden zu entwickelnden Werkzeuge zu gewährleisten, ist es notwendig, dass diese sowohl mit dem GroupProcess-Projekt als auch mit der Organization Database von PAVONE Enterprise Office kommunizieren. Es ist also notwendig einerseits eine Kommunikationsmöglichkeit zwischen zwei Java-Applikationen zu finden und andererseits eine Zugriffsmöglichkeit auf Lotus Notes-Datenbanken, also einer Nicht-Java-Applikation, zu entwickeln. In den Kapiteln 3.5 und 3.6 wurden Alternativen der Kommunikation innerhalb von Java-Komponenten und zwischen Java und Nicht-Java-Applikationen aufgezeigt. Diese Alternativen werden in diesem Kapitel gegeneinander abgewogen, um einen optimalen Nachrichtenaustausch zwischen den einzelnen Komponenten zu gewährleisten.

4.3.1 Anbindung des Organizationchart- / WorkgroupViewers an das Ad-hoc-Workflow-Managementwerkzeug

Wie bereits in Kapitel 4.2 erwähnt ist eine Implementierung der beiden Viewer in Java vorgesehen. Bei der Anbindung der neuen Komponenten an das Ad-hoc-Workflow-Modellierungswerkzeug sind neben den speziellen Techniken des Informationsaustausches zwischen Java-Applikationen zusätzlich auch die in Kapitel 3.5 erwähnten Kommunikationsmöglichkeiten zwischen Java- und Nicht-Java-Applikationen einsetzbar.

Die beiden Viewer stellen eine Erweiterung des Ad-hoc-Workflow-Modellierungswerkzeuges dar. Insofern ist ein Szenario, in dem der Organizationchart- / WorkgroupViewer vom GroupProcess-Projekt getrennt in einem verteilten System zum Einsatz kommt nicht sinnvoll. Diese Tatsache schließt zwar keine der

beschriebenen Kommunikationstechniken aus, allerdings sind jene, die aufgrund ihrer Flexibilitätsausrichtung einen hohen Overhead produzieren nicht geeignet, da sie die Performance des Gesamtsystems unnötig belasten würden. Zu diesen Techniken zählen insbesondere CORBA, RMI und COM, wobei COM wegen seiner Abhängigkeit von Microsofts Betriebssystemen bereits von vorneherein von einer weiteren Betrachtung auszuschließen ist.

Aufgrund der Tatsache, dass bei den weiteren verfügbaren Kommunikations-möglichkeiten über temporäre Dateien oder Datenbanken, die datenerhaltende Komponente ständig überprüfen muss, ob neue Daten vorliegen, würde eine solche Lösung einen hohen Netzwerkverkehr und eine hohe Ressourcenauslastung hervorrufen. Daher werden diese Lösungsalternativen in diesem Szenario nicht als relevant betrachtet.

Die Kommunikation über Interfaces ist eine reine Java-Lösung, welche die oben genannten Nachteile nicht mit sich bringt. Des Weiteren entsteht durch die Verwendung von Interfaces eine Schnittstelle für die Datenübergabe auch zu anderen Applikationen, welche nicht zwangsläufig dem GroupProcess-Projekt angehören müssen. Daher ist eine solche Kommunikation welche Java-Interfaces implementiert sinnvollerweise anzustreben.

4.3.2 Anbindung des GroupProcess-Projektes an Lotus Notes-Datenbanken

Im Falle des Informationsaustausches des GroupProcess-Projektes mit den Lotus Notes-Datenbanken sind die Techniken, die temporäre Dateien oder Datenbanken einsetzen, aus den bereits im vorhergehenden Unterkapitel genannten Gründen zu vermeiden. Da Lotus Notes keine Java-Applikation darstellt, entfallen in dieser Betrachtung sämtliche Kommunikationstechniken, die nur zwischen Java-Anwendungen einsetzbar sind.

Dadurch sind von den insgesamt dargestellten Kommunikationsmöglichkeiten nur noch CORBA und die von Lotus Notes bereitgestellten speziellen Java-Klassen zur Implementation relevant. Die von Lotus Notes bereitgestellte Java-Unterstützung basiert bei verteilter Anwendung auf der CORBA Technologie, ohne allerdings deren gesamte Flexibilität anzubieten. Diese Tatsache führt dazu, dass durch einen Einsatz der Lotus Notes-Java-Klassen der Gesamtumfang des GroupProcess-Projektes und der Overhead

reduziert wird. Dies ist insbesondere interessant, da das GroupProcess-Projekt auch im Internet einsetzbar ist, wo aus Gründen der Performance auf möglichst geringe Datentransfervolumen zu achten ist. Daher sollten bei einer Implementation des Kommunikationsmoduls des GroupProcess-Projektes die Lotus Notes-Java-Klassen zum Einsatz kommen.

5 Eine Java-basierte Lösung für einen Organizationchart- und einen WorkgroupViewer und die Anbindung an Lotus Notes

Im Kapitel 4 wurden Lösungsansätze für die Implementation der beiden Benutzungsschnittstellen des Organizationchart- und WorkgroupViewers sowie deren Anbindung an Lotus Notes und insbesondere an die vorhandene Organisationsdatenbank von PAVONE dargestellt. Den Anfang dieses Kapitels bildet die Vorstellung der beiden Prototypen für die Arbeitsgruppen- und Abteilungsstrukturvisualisierung. Im weiteren Verlauf wird eine Implementation der Lotus Notes-Kommunikationsmodule für diese beiden Komponenten dargestellt. Den Abschluss dieses Kapitels bildet eine Erläuterung der Einsatzmöglichkeiten dieser Prototypen.

5.1 Prototypen des OrganizationchartViewers und des WorkgroupViewers

Bei der Realisation der beiden Werkzeuge zur Visualisierung der Arbeitsgruppen- und Abteilungsstrukturen gingen insbesondere die in Kapitel 3 erörterten Überlegungen über die Gestaltung von Benutzungsschnittstellen ein.

Die Implementation der Werkzeuge erfolgte in Java, um dem Anspruch an die Plattformunabhängigkeit des gesamten GroupProcess-Projektes Rechnung zu tragen. Des Weiteren ergab sich für die Kommunikation mit Lotus Notes der Vorteil, dass die Lotus Notes eigenen Java-Klassen zur Kommunikation der beiden Tools mit Lotus Notes eingesetzt werden konnten.

5.1.1 Dialogmodellierung

Die folgende Abbildung zeigt den dialogtechnischen Ablauf innerhalb des GroupProcess-Projektes mit Einbindung der beiden Prototypen für die Abteilungsstruktur- und die Arbeitsgruppenansicht.

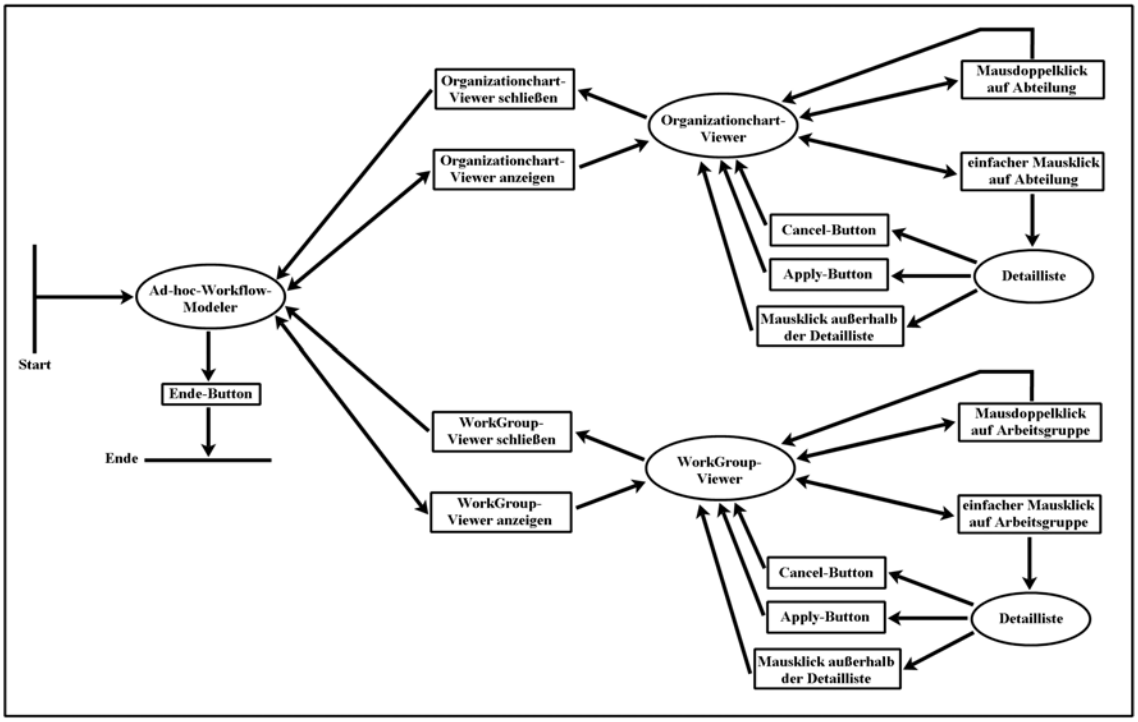

Abbildung 26: Dialognetz der Werkzeuge nach Janssen

Das in dieser Abbildung dargestellte Janssen-Netz zeigt den Ablauf der Dialoge im GroupProcess-Projekt. Ausgehend vom Ad-hoc-WorkflowModeler wurden allerdings nur die Dialoge in den für diese Arbeit relevanten Werkzeugen modelliert. Gut zu erkennen ist die Tatsache, dass der Dialogablauf für beide Viewer nahezu identisch ist, um dem Anspruch an die in Kapitel 3.4.2 beschriebene Erwartungskonformität gerecht zu werden.

Die beiden Werkzeuge sind aus dem Ad-hoc-Workflow-Modelierungswerkzeug des GroupProcess-Projektes über Buttons aufzurufen. Sie enthalten die aktuell in einer Organization Database hinterlegten Gruppen- und Abteilungsstrukturen. Durch die Auswahl einer gesamten Abteilung mit einem Doppelklick auf diese im OrganizationchartViewer werden die entsprechenden Abteilungsdaten an den Ad-hoc-WorkflowModeler übergeben. Genauso verhält es sich mit den Daten einer Arbeitsgruppe im WorkgroupViewer. Durch die Auswahl eines Gruppensymbols im Arbeitsgruppenwerkzeug oder einer Abteilung im Abteilungsstrukturwerkzeug mit einem einfachen Mausklick wird die Personenübersicht der entsprechenden Organisationseinheit angezeigt. In dieser Liste kann eine Auswahl bezüglich der

Personen getroffen werden. Diese Selektion kann entweder durch Anwahl des „Apply"-Buttons an den Ad-hoc-WorkflowModeler übermittelt oder durch drücken des „Cancel"-Buttons annulliert werden. Die beiden Applikationen können durch die Betriebssystemeigene Funktion des Fensterschließens beendet werden.

5.1.2 Optische Gestaltung

Die optische Gestaltung der beiden Werkzeuge erfolgte nach den in Kapitel 4 erstellten Anforderungen. In beiden Werkzeugen wurde die Gestaltung der Organisationseinheiten an die in den gängigen Betriebssystemen übliche 3D-Optik angepasst. Zu diesem Zweck werden sowohl die Abteilungs- als auch die Arbeitsgruppensymbole sowie die dazugehörigen Personenlisten mit einem Rahmen umgeben, der im Falle eines Wechsels der Auswahl dieses dem Benutzer durch einen entsprechenden 3D-Effekt kenntlich macht.

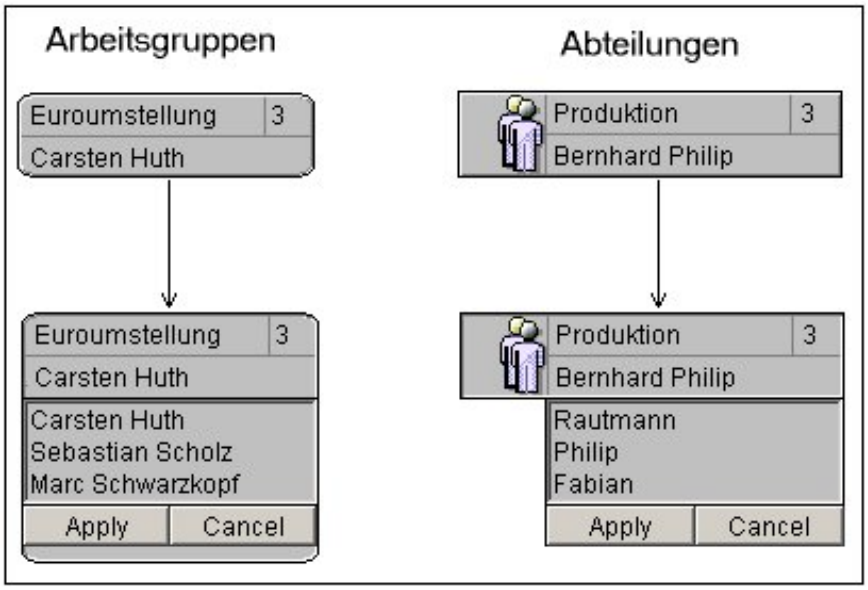

Abbildung 27: Der 3D Effekt bei den Organisationseinheiten

Bei beiden Werkzeugen wird im Falle der durch einen Mausklick ausgeführten Auswahl einer Organisationseinheit eine Auswahlliste mit den in dieser Einheit vorhandenen Personen angezeigt. Innerhalb dieser Liste können eine, aber auch mehrere Personen selektiert werden. Auch die Auswahlliste ist an das Konzept der 3D-Visualisierung angepasst, um die aktuelle Auswahl hervorzuheben.

Um dem Anspruch an Selbsterklärungsfähigkeit zweiten Grades gerecht zu werden, ist in beiden Werkzeugen eine Hilfeleiste integriert, die beim Berühren eines Objektes mit dem Mauszeiger eine passende Onlinehilfe anzeigt.

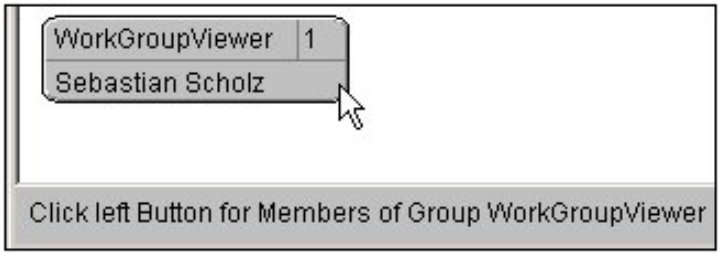

Abbildung 28: Die Hilfeleiste

5.1.2.1 Die Optische Gestaltung des OrganizationchartViewers

Die Benutzeroberfläche des OrganizationchartViewers ist optisch stark an die des OrganizationModelers von PAVONE angepasst. Die folgende Abbildung zeigt einen Screenshot des OrganizationchartViewers.

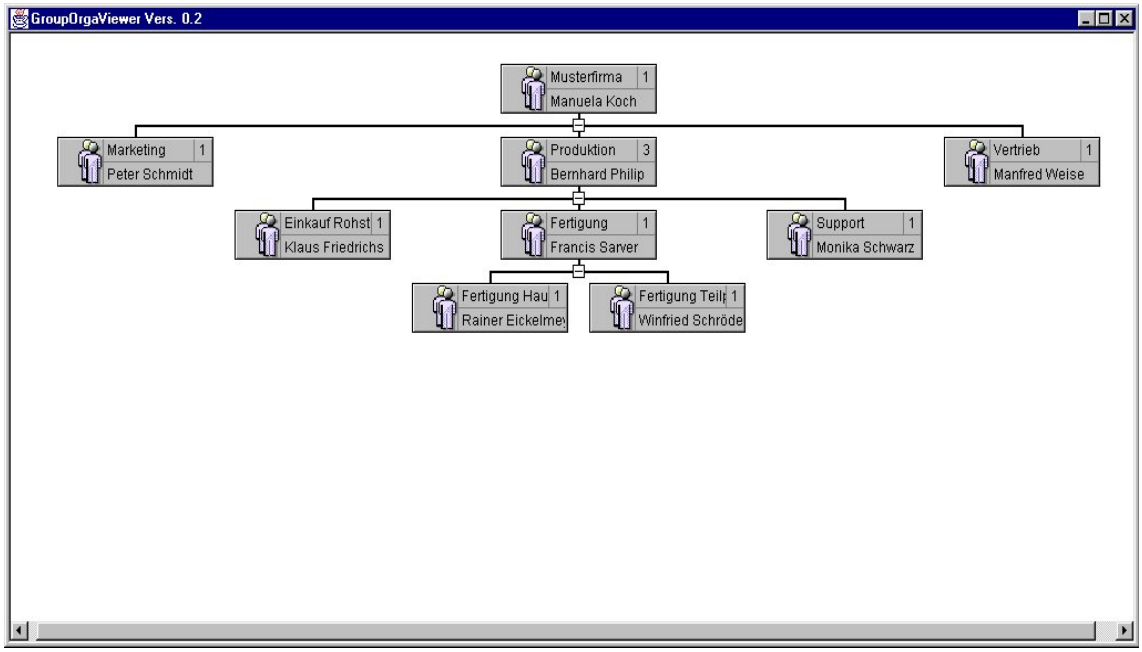

Abbildung 29: Der OrganizationchartViewer

Die hier sichtbaren Abteilungssymbole vor der jeweiligen Abteilung sind Bestandteil einer parallel zu dieser Arbeit durchgeführten Seminararbeit und werden im Folgenden nicht weiter betrachtet.

Innerhalb der einzelnen Abteilungen wird zum einen der Name der Abteilung angezeigt und zum anderen der Leiter der jeweiligen Abteilung. Zusätzlich erscheint zur einfacheren Übersicht neben dem Abteilungsnamen die Anzahl der Mitarbeiter der betreffenden Abteilung.

Die einzelnen Abteilungen im OrganizationchartViewer sind zur Kennzeichnung der Hierarchiebeziehungen mit entsprechenden Verbindungslinien aneinander gebunden. Einzelne Teilbäume innerhalb der Hierarchie können durch die Betätigung der Anzeigen- und Verbergen-Symbole in Form eines Plus- bzw. Minuszeichens im Mittelpunkt der Verbindungslinie ein- und ausgeblendet werden. Somit kann der Hierarchiebaum auf den für den Benutzer relevanten Bereich reduziert angezeigt werden, was die folgende Abbildung verdeutlicht.

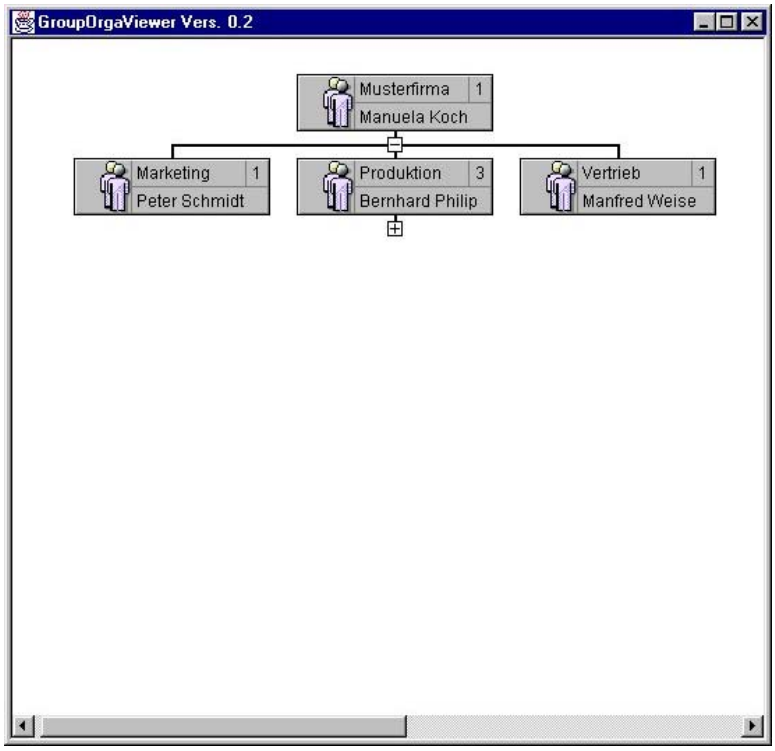

Abbildung 30: Reduzieren des Hierarchiebaumes

5.1.2.2 Die Optische Gestaltung des WorkgroupViewers

Beim WorkgroupViewer sind die einzelnen Arbeitgruppen mangels hierarchischer Beziehungen nicht miteinander verbunden und können an unterschiedlichen Stellen innerhalb der graphischen Darstellung platziert sein. Die Platzierung der

Arbeitsgruppensymbole auf der Oberfläche werden durch das Ad-hoc-Workflow-Modellierungswerkzeug des GroupProcess-Projektes gesteuert. Dazu werden von dort aus die Koordinatenpunkte der Arbeitsgruppensymbole an den WorkgroupViewer übergeben. Die Koordinaten werden in den Daten der Arbeitsgruppe, innerhalb der Organisation Database von PAVONE, durch den Ad-hoc-WorkflowModeler gespeichert.

Wie auch beim OrganizationchartViewer wird innerhalb des Arbeitsgruppensymbols der Name der betreffenden Arbeitsgruppe, der Gruppenleiter sowie die Anzahl der zu der jeweiligen Arbeitsgruppe gehörigen Personen angezeigt. Die folgende Abbildung zeigt mögliche Anordnungen innerhalb des WorkgroupViewers.

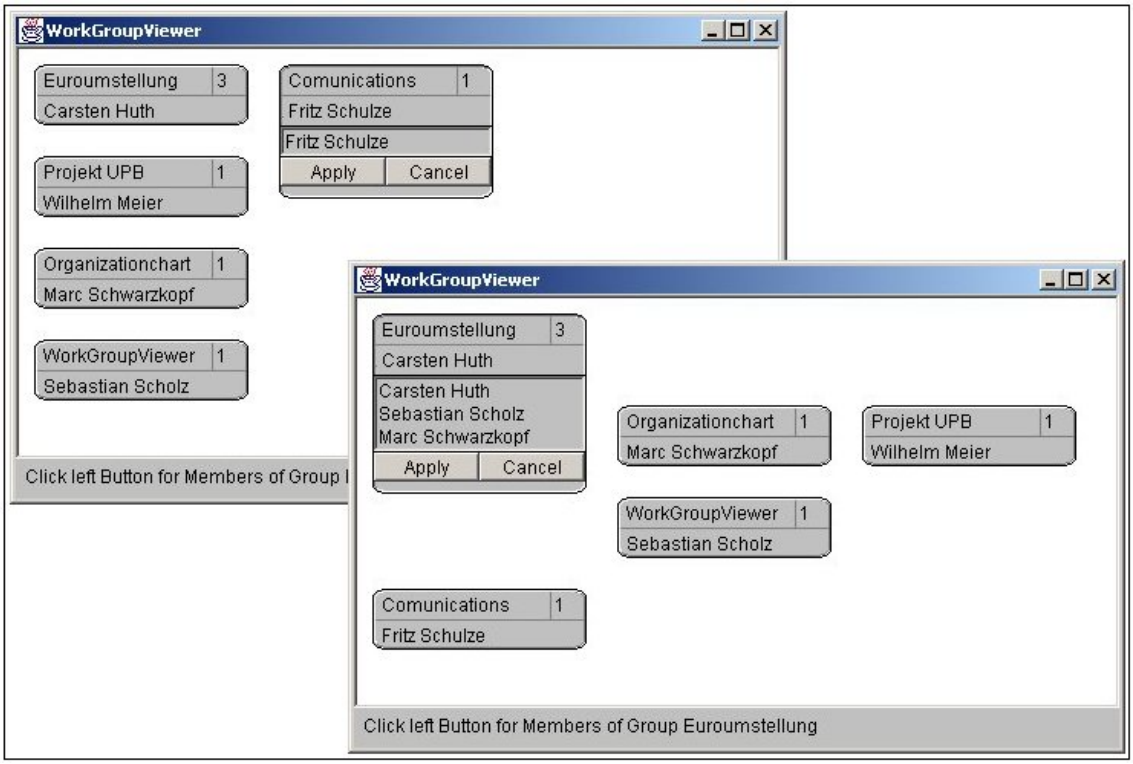

Abbildung 31: Verschiedene Anordnungen beim WorkgroupViewer

Die optische Darstellung des WorkgroupViewers ist mehr an den Organizationchart-Viewer angelehnt, um eine möglichst konsistente Gestaltung und Bedienung über sämtliche Module des GroupProcess-Projektes zu erreichen. Durch Abrundung der Ecken der Arbeitsgruppensymbole ist beim WorkgoupViewer ein Kompromiss zwischen der einheitlichen Gestaltung und der für Arbeitsgruppensymbole üblichen Optik eingegangen worden.

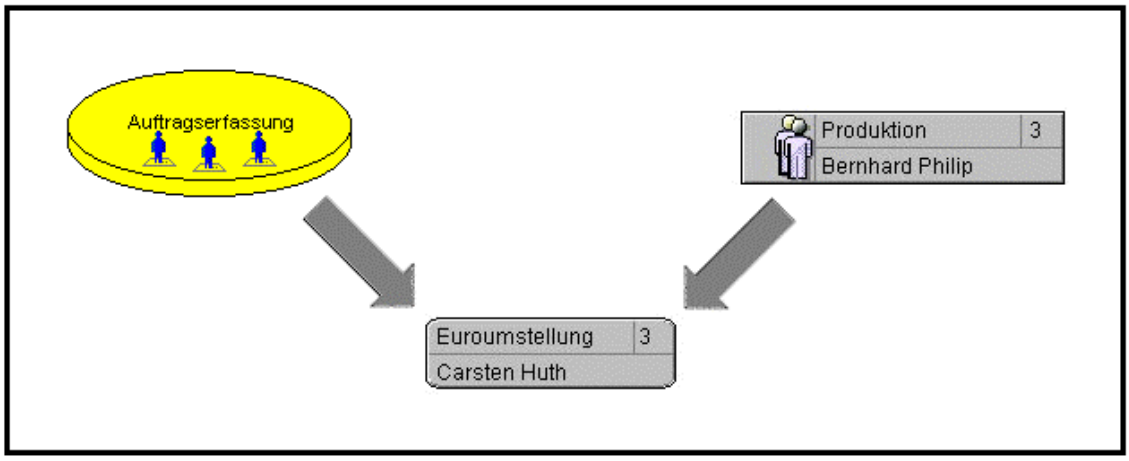

Abbildung 32: Die Darstellung der Arbeitsgruppen

5.2 Architektur des OrganizationchartViewers, des WorkgroupViewers und der Kommunikation

Die Implementation der beiden Viewer wurde wie schon bereits erwähnt in Java vorgenommen. Im Bereich des Softwareengineerings kamen hierbei die Grundsätze der objektorientierten Programmierung zum Einsatz. Die beiden neuen Viewer sowie das Kommunikationsmodul zur Kommunikation mit Lotus Notes sind Bestandteil des GroupProcess-Projektes. Aus diesem Grund wurden diese Teilmodule in der bereits bestehenden Package-Struktur des GroupProcess-Projektes integriert. Die folgende Abbildung zeigt diese Struktur.

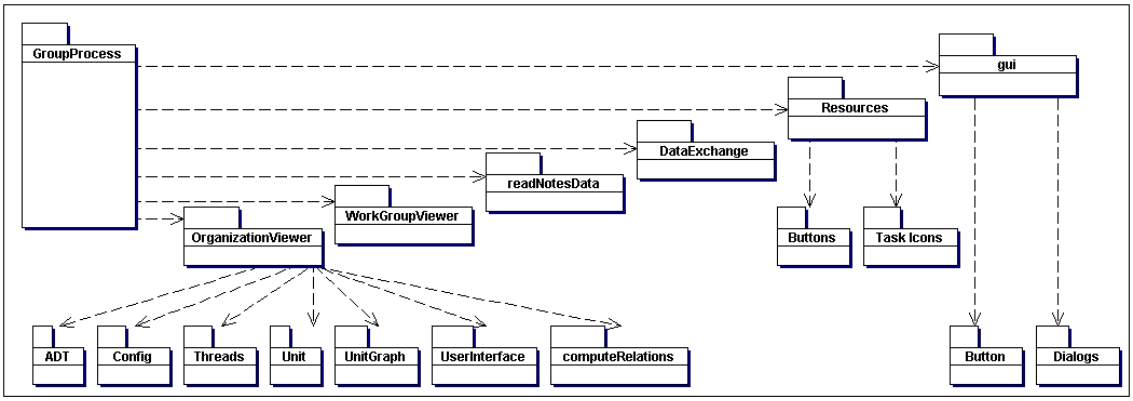

Abbildung 33: Die Packagestruktur des GroupProcess-Projektes

In das GroupProcess-Projekt wurden durch diese Arbeit die zusätzlichen Packages OrganizationchartViewer, WorkgroupViewer und readNotesData eingefügt. Die beiden ersteren Pakete enthalten sämtliche Klassen und Interfaces, die zur Darstellung,

Funktion und zur Kommunikation der beiden Viewer mit dem Ad-hoc-Workflow-Modellierungswerkzeug notwendig sind. Das readNotesData-Paket enthält sämtliche, zur Kommunikation mit Lotus Notes notwendigen Klassen.

5.2.1 Die Architektur des OrganizationchartViewers

Wie in Abbildung 33 zu erkennen ist, besteht das OrganizationchartViewer-Modul aus mehreren Teilpaketen. Innerhalb dieser Pakete sind verschieden Klassen zur optischen Darstellung, zur Interaktion mit dem Benutzer und zur Berechnung der Hierarchiebeziehungen unter den Abteilungen enthalten. Die Implementierungsbeziehungen zwischen Klassen und Interfaces sowie die Vererbungsstruktur zwischen den Klassen werden in der folgenden Abbildung dargestellt.

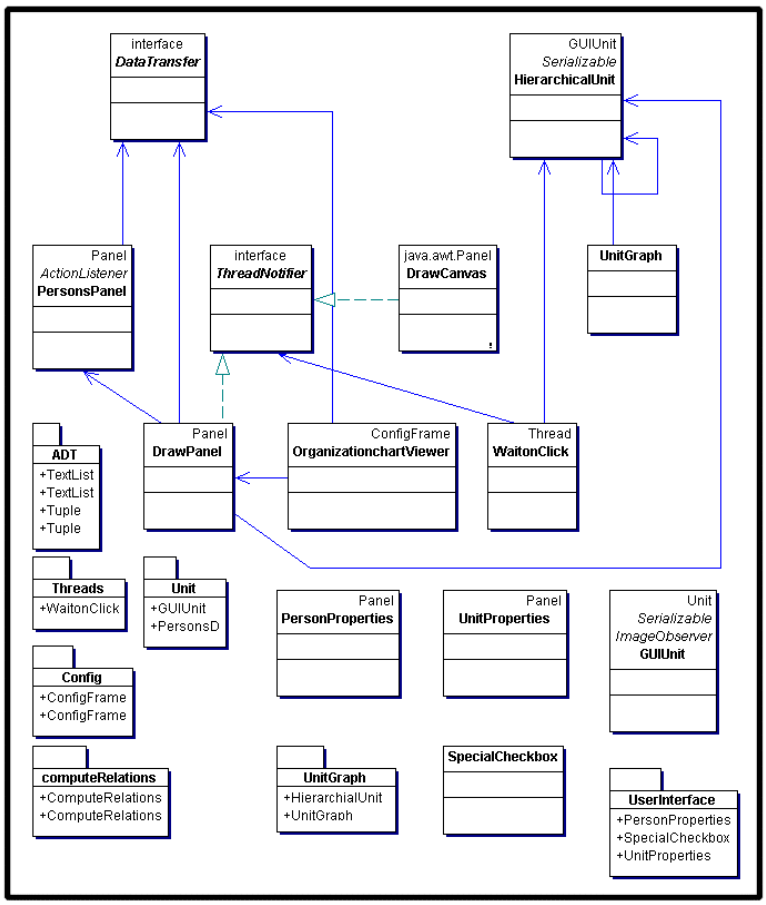

Abbildung 34: Klassendiagramm des OrganizationchartViewers

Die Klasse OrganizationchartViewer wird von der Klasse ConfigFrame abgeleitet und spielt die zentrale Rolle innerhalb des Moduls zur Darstellung der Abteilungshierarchie. Hier wird zum einen die gesamte Funktionalität der interaktiven Dialoge verwaltet und zum anderen der graphische Aufbau der Oberfläche koordiniert. Die Basisklasse ConfigFrame enthält die dazu nötigen Attribute, welche von der aufrufenden Applikation über entsprechende Methoden gesetzt werden können.

Durch einen Aufruf aus der Klasse OrganizationchartViewer errechnet die Klasse computeRelations aus den aus Lotus Notes transferierten Daten eine Baumstruktur der Hierarchiebeziehungen. Diese Struktur wird in Instanzen der Klasse HierarchialUnit abgelegt. Die Basis von HierarchialUnit bildet die Klasse GUIUnit, welche die zur graphischen Anzeige der Abteilung erforderlichen Daten, wie z.B. die Bildschirmpositionen enthält. Die Klasse Unit bildet wiederum die Grundlage für GUIUnit und hält in Attributen sämtliche Daten, wie z.B. Abteilungsnamen und Mitglieder, die zur Beschreibung der entsprechenden Abteilung notwendig sind.

Die Klasse PersonsD ist zuständig für die Anzeige der in den Abteilungen enthaltenen Mitarbeiter. Um zur Darstellung der Personen einen zusätzlichen Bereich auf der Oberfläche erzeugen zu können, ist sie von der Klasse Panel aus dem Java eigenen AWT-Paket abgeleitet. Eine Instanz der Klasse PersonsD wird durch einen einzelnen Mausklick auf eine Abteilung erzeugt und in die graphische Oberfläche eingefügt. Innerhalb des erzeugten Panels befinden sich neben einer Listbox zur Anzeige und Auswahl der betreffenden Personen noch zwei Buttons für die Auslösung des Datentransfers zum Ad-hoc-Workflow-Modellierungswerkzeug und zum Abbruch der Aktion.

5.2.2 Die Architektur des WorkgroupViewers

Die Klassenstruktur des WorkgroupViewers ist im Gegensatz zu der des OrganizationchartViewers übersichtlicher. Dies resultiert aus der Tatsache, dass im Bereich der Darstellung von Arbeitsgruppen keine hierarchischen Beziehungen existieren. Dadurch entfallen beim WorkgroupViewer sowohl die gesamte Berechnung des Hierarchiebaums und dessen Speicherung, als auch die Algorithmen zur Ein- und Ausblendung von Teilbäumen.

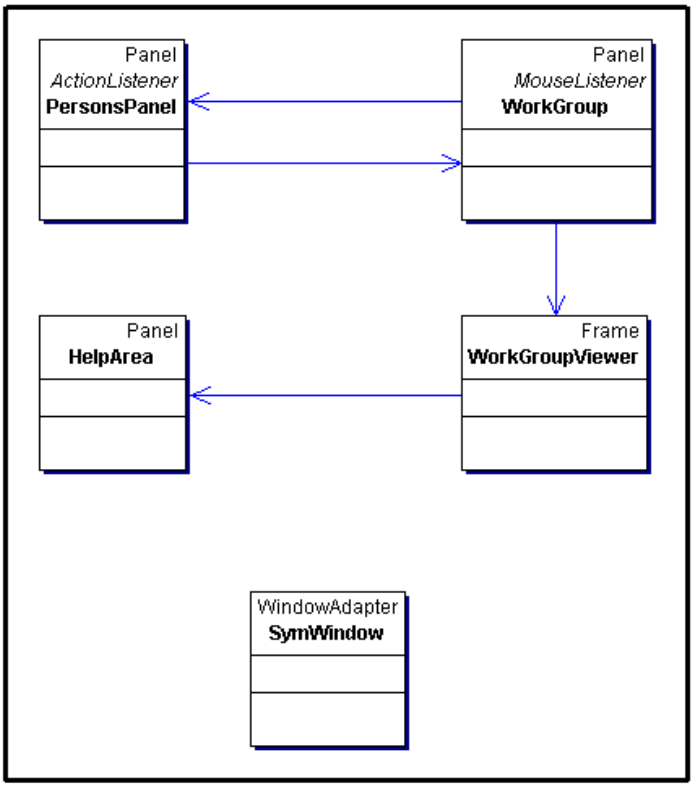

Abbildung 35: Die Klassenstruktur des WorkgroupViewers

Das in der obigen Abbildung dargestellte Klassendiagramm zeigt die Strukturierung des WorkgroupViewers. Auch hier existiert, wie beim OrganizationchartViewer eine zentrale Klasse, welche die komplette Funktionalität und den graphischen Aufbau der Oberfläche übernimmt. Hierbei handelt es sich um die Klasse WorkgroupViewer. Die Klasse PersonsPanel wurde zum größten Teil von der Klasse PersonsD des OrganizationchartViewers übernommen, wobei die Methoden zur graphischen Darstellung der Personenübersicht an die im WorkgroupViewer verwendete Darstellungsweise mit abgerundeten Ecken angepasst wurde. Um eine Eigenständigkeit der einzelnen Pakete zu gewährleisten, wurde bei der Klasse PersonsPanel bewusst auf eine Vererbung aus PersonsD verzichtet.

In der Klasse Workgroup werden sowohl die aus Lotus Notes erhaltenen Daten der einzelnen Arbeitsgruppen, als auch die zur graphischen Darstellung benötigten Informationen gespeichert. Zusätzlich verfügt sie über die notwendigen Methoden zum optischen Aufbau der Workgroup-Symbole samt aller Details.

HelpArea bezeichnet die Klasse, welche für die Darstellung der integrierten Onlinehilfe zuständig ist. Die Anzeige der Hilfetexte erfolgt am unteren Rand der graphischen Oberfläche und ist abhängig von der betreffenden Arbeitsgruppe, auf welcher der Mauszeiger positioniert ist. Insofern ist es möglich, in jeder Instanz der Klasse Workgroup einen für sie spezifischen Hilfetext zu hinterlegen.

5.2.3 Der Aufbau der verwendeten Kommunikationslösung

Die in Kapitel 4 erstellten Lösungsstrategien für die Kommunikation zwischen Lotus Notes und dem GroupProcess-Projekt sowie die zwischen den beiden Viewern und dem Ad-hoc-Workflow-Modellierungswerkzeug wurden bei der Implementierung der Kommunikation verwendet. Aufgrund der unterschiedlichen Kommunikationswege ist eine getrennte Darstellung der realisierten Lösungen notwendig.

5.2.3.1 Kommunikation zu Lotus Notes

Zur Realisierung des Informationsaustausches mit Lotus Notes wurde eine gesondertes Package in der Hierarchie des GroupProcess-Projektes geschaffen, welches die von Lotus Notes bereitgestellten Java-Klassen benutzt. Dieses ist im Hierarchiebaum unter readNotesData zu finden.

Die von Lotus Notes bereitgestellten Java-Klassen werden innerhalb der Klasse OrganizationData dazu genutzt, die PAVONE Organisationsdatenbank zu öffnen und die dort gespeicherten Daten der Personen und Organisationseinheiten zu gewinnen. Die gewonnenen Daten werden in Instanzen der Klassen Person, Unit und Group abgelegt, welche wiederum in OrganizationData in Vectoren[86] gespeichert und über get-Methoden[87] abrufbar sind.

5.2.3.2 Kommunikation mit dem Ad-hoc-WorkflowModeler

Die Kommunikation mit dem Ad-hoc-Workflow-Modellierungswerkzeug wurde, wie bereits im Lösungsansatz in Kapitel 4.3.1 erwähnt, durch den Einsatz von Interfaces

[86] Vectoren sind unter Java doppelt verkettete Listen beliebiger Objekte

[87] get-Methoden werden in objektorientierten Programmiersprachen dazu verwendet, den Wert von Attributen eines Objektes auszulesen, ohne einen direkten Zugriff auf diese zu ermöglichen.

realisiert. Dazu wurde die Interfaceklasse DataTransfer erstellt, welche die abstrakten Methoden zur Übergabe der Daten enthält. Dieses Interface wird von der Klasse FrameActionListener des Ad-hoc-Workflow-Modellierungswerkzeuges implementiert. Dadurch sind die Vorraussetzungen für die Übertragung der Daten gegeben.

Wenn der Benutzer im OrganzationchartViewer eine Abteilung auswählt, so wird die innerhalb des FrameActionListeners implementierte Methode UnitSelected aufgerufen. Beim Aufruf dieser Methode wird als Parameter das ausgewählte Unit-Objekt übergeben. Dadurch kommt die Klasse FrameActionListener an die Informationen der Abteilung und kann diese intern weiterverarbeiten.

Diese Vorgehensweise bei der Übertragung der Daten aus dem Organizationchart-Viewer wird auch beim Workgroupviewer angewandt. Das Interface DataTransfer stellt für die Übertragung aus dem WorkgroupViewer sowie auch für die Übertragung ausgewählter Personen der beiden Viewer entsprechende abstrakte Methoden bereit.

5.3 Flexibilität und Einstellungsmöglichkeiten der Werkzeuge

In der Entwicklungsphase der Werkzeuge wurde bereits auf hohe Flexibilität und Anpassungsfähigkeit Wert gelegt. Daher wurden in der Implementierungsphase Module geschaffen, die es zum einen ermöglichen, die Werkzeuge auch in anderen Umgebungen außerhalb des GroupProcess-Projektes einzusetzen und es zum anderen realisieren, die optische Darstellung der Viewer anzupassen.

5.3.1 Optische Anpassung der Komponenten

Um den verschiedenen Anforderungen bei der graphischen Ausgabe der Abteilungsstruktur gerecht zu werden, wurden innerhalb der ConfigFrame-Klasse Einstellungsmöglichkeiten bezüglich der Größen der Abteilungssymbole und der Gesamtanwendung sowie der Abstände zwischen den einzelnen Symbolen implementiert. Dadurch können bei komplizierten hierarchischen Strukturen Abstände und Größen derart an die individuellen Systemvoraussetzungen angepasst werden, dass eine Darstellung auch solcher Strukturen ohne den Einsatz der Scrollleisten ermöglicht werden kann. Die Einstellungsmöglichkeiten werden zusätzlich durch die Möglichkeit, den Hierarchiebaum vertikal oder horizontal anzeigen zu lassen abgerundet.

Zu diesem Zweck wurden in der ConifgFrame-Klasse entsprechende Methoden bereitgestellt, die es dem aufrufenden Programm ermöglichen, die Anpassungen an den Umgebungsattributen vorzunehmen. Sollte die aufrufende Applikation keine Angaben zu der Darstellung übergeben, so werden automatisch die voreingestellten Standardwerte zur Darstellung des OrganizationchartViewers benutzt. Die gleichen Anpassungsmöglichkeiten sind auch beim WorkgroupViewer implementiert.

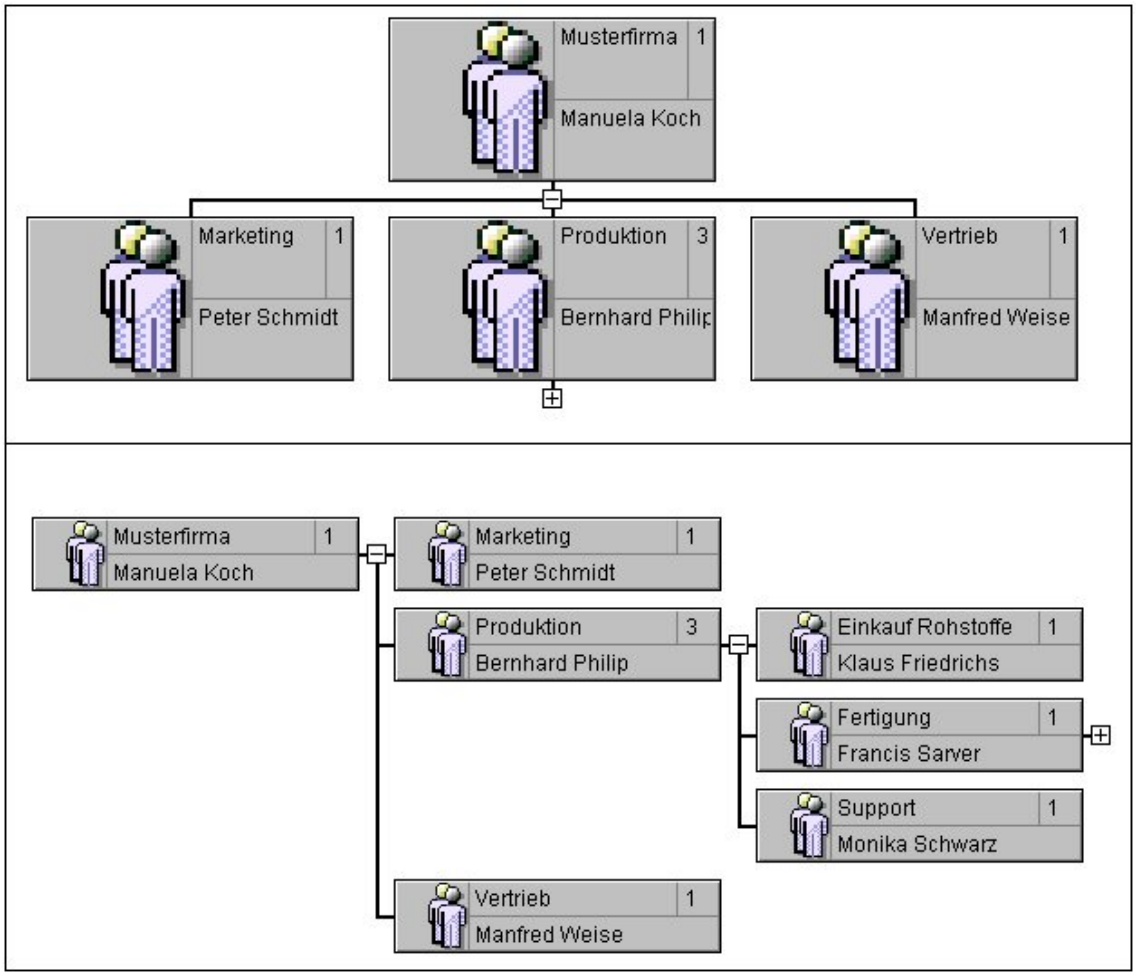

Abbildung 36: Verschiedene Darstellungsmöglichkeiten des OrganizationchartViewers

Abbildung 36 zeigt mögliche Darstellungen in Abhängigkeit von den Einstellungen in der ConfigFrame-Klasse. Diese Einstellungen wurden der Abteilungsstruktur der Organisation angepasst.

5.3.2 Flexibilität der Werkzeuge

Bei der Entwicklung des gesamten GroupProcess-Projektes wurde Wert auf eine hohe Flexibilität gelegt. Daher wurden auch die beiden zusätzlichen Komponenten OrganizationchartViewer und WorkgroupViewer mit Hilfe der Java-Technologie implementiert.

Darüber hinaus wurde durch den Einsatz von Interfaces zur Kommunikation mit der aufrufenden Komponente die Möglichkeit geschaffen, den OrganizationchartViewer und den WorkgroupViewer in Kombination mit beliebigen anderen Java-Klassen einzusetzen, die das Kommunikationsinterface DataTransfer implementieren. Die Kommunikation mit Lotus Notes über die bereitgestellten Java-Klassen ermöglicht einen schlanken und effizienten Informationsaustausch. Da diese Klassen im verteilten Fall auf der CORBA-Technologie basieren, stehen dem weltweiten Einsatz des GroupProcess-Projektes mit der Nutzung des Internets als Kommunikationsmedium alle Wege offen.

Die Kommunikation mit Lotus Notes wurde im Package readNotesData realisiert. Dieses Modul kann jederzeit auf andere Workflow-Management-Systeme, welche nicht die Lotus Notes-Java-Klassen bereitstellen, angepasst werden. Die Funktionalität der Module würde dadurch nicht beeinträchtigt.

6 Ausblick

Die aktuellen Entwicklungen, insbesondere bei der kontinuierlichen Verbreitung des Internets und der zunehmenden Globalisierung stellen Organisationen vor neue Herausforderungen bei der organisationsinternen und -externen Kommunikation. Das Konzept der virtuellen Organisationen wird auch in Zukunft zunehmend an Bedeutung gewinnen. Lotus Notes als Groupware-Plattform bietet eine passende Unterstützung solcher Unternehmensstrukturen, insbesondere durch die Möglichkeit des verteilten Einsatzes.

Organisationen benötigen aufgrund ihres zunehmenden Wachstums Systeme, die sowohl die Ablaufmodellierung, als auch die Aufbaustrukturmodellierung unterstützen. Dieser Bereich wird von der Firma PAVONE bereits mit dem Produkt Enterprise Office unterstützt. PAVONE bietet zur Modellierung der virtuellen Unternehmung schon einen Großteil der geforderten Funktionalität. Im Bereich Ad-hoc-Workflow-Management wird allerdings zunehmend auf einfache und schnelle Modellierung solcher Workflows gesetzt. Das GroupProcess-Projekt soll diesen Bereich ergänzen und gleichzeitig flexibel auf zukünftige Bedürfnisse anpassbar sein. Für die schnelle und komfortable Auswahl von Organisationseinheiten wurden im Rahmen dieser Arbeit prototypische Werkzeuge entwickelt, welche die Erstellung von Ad-hoc-Workflows erleichtern.

Diese Prototypen bilden in diesem Stadium bereits Grundfunktionalitäten zur Auswahl von Personen und Organisationseinheiten an. In Zukunft sind jedoch mehrere Anpassungs- und Erweiterungsalternativen denkbar.

Sicherlich ist in zukünftigen Versionen von Lotus Notes eine Unterstützung der aktuellen Java-Version 1.3 sowie auch der Java-Swing-Klassen anzunehmen, wodurch eine Anpassung der Oberflächen der entwickelten Prototypen auf das sogenannte „Look and Feel“[88] möglich würde. Zusätzlich ist im Rahmen einer Seminararbeit bereits die Erweiterung der Symbole für die Organisationseinheiten um Icons geplant. Dadurch wird eine einfachere und schnellere Identifizierung der einzelnen

[88] Java-Swing erweitert Java um Komponenten zur Modellierung einer einheitlichen Optik, die mit dem Namen „Look and Feel“ betitelt wird.

Organisationseinheiten ermöglicht. Eine solche Darstellung wäre auch innerhalb der Personenlisten denkbar, in denen dann zusätzlich zu den Namen Bilder der betreffenden Personen angezeigt werden könnten.

Eine Erweiterung des WorkgroupViewers um die freie Positionierung der einzelnen Workgroups durch einfaches Ziehen mit der Maus würde die Bedienung dieses Werkzeuges komfortabler gestalten. Die Positionen der Arbeitsgruppen auf der Oberfläche bräuchten dann nicht innerhalb der Organization Database von PAVONE festgelegt werden, sondern könnten direkt innerhalb des WorkgroupViewers zugeordnet werden.

7 Zusammenfassung

Im Rahmen dieser Arbeit wurden zunächst grundlegende Konzepte des Workflow-Managements und die daraus resultierenden Anforderungen an die Ad-hoc-Workflow-Modellierung erörtert. Das Konzept dieser Ad-hoc-Workflow-Modellierung ist grundlegender Bestandteil des GroupProcess-Projektes, welches durch diese Arbeit erweitert wurde. Diese Erweiterung besteht darin, dem Ad-hoc-Workflow-Modellierungswerkzeug Hilfsmittel zur Verfügung zu stellen, die es den Benutzern erleichtern, Organisationseinheiten oder Personen aus einer virtuellen Organisationsstruktur auszuwählen. Die Auswahl erfolgt über zwei Werkzeuge: den OrganizationchartViewer zur Anzeige und Selektion von Abteilungsstrukturen und den WorkgroupViewer, welcher die Darstellung und Selektion von Arbeitsgruppen einer Organisation realisiert.

Um die Bedienung dieser beiden zusätzlichen Applikationen benutzergerecht zu gestalten, wurden grundlegende Aspekte der Benutzungsschnittstellenmodellierung und Dialoggestaltung in dieser Arbeit erläutert. Des Weiteren wurden allgemein übliche Darstellungsweisen für Aufbauorganisationen aufgezeigt, um daraufhin eine entsprechende Symbolik für Abteilungen und Arbeitsgruppen entwickeln zu können.

Der Einsatz der Java-Technologie bei der Implementierung des GroupProcess-Projektes erforderte eine intensive Betrachtung dieser Technologie, da die neuen Auswahlsysteme in diesem Projekt integriert werden sollten. Zusätzlich ergaben sich dadurch und durch die geforderte Anbindung an die Groupware-Umgebung Lotus Notes diverse Kommunikationsproblematiken, welche es erforderlich machten, mögliche Alternativen zu deren Lösung zu analysieren.

Bei der Findung eines Lösungsansatzes wurde eine Kompromiss zwischen der Darstellungsweise bereits existierender Organisationsmodellierungswerkzeuge und einem einheitlichen Erscheinungsbild innerhalb des GroupProcess-Projektes eingegangen. Zusammen mit weiteren Kriterien der Modellierung von Benutzschnittstellen und Dialogen wurde ein Konzept für die Erstellung der graphischen Oberflächen geschaffen.

Im Bereich der Kommunikation zwischen den beteiligten Komponenten wurde eine Lösung gefunden die sowohl der Forderung nach Flexibilität als auch dem Anspruch auf eine, beim verteilten Einsatz der Systeme notwendige Effizienz gerecht wird. Diese Lösung nutzt zur Kommunikation innerhalb des GroupProcess-Projektes die von Java bereitgestellte Interface-Technologie. Als Schnittstelle zu Lotus Notes dienen spezielle Java-Klassen, die auf der CORBA-Technologie basieren.

Die gefunden Lösungsansätze wurden praktisch umgesetzt und stehen dem GroupProcess-Projekt zum Einsatz zur Verfügung.

8 Literaturverzeichnis

Abts 1998:

Grundkurs Wirtschaftsinformatik: Eine kompakte und praxisorientierte Einführung / Dietmar Abts / Wilhelm Mülder. – 2. Auflage, Braunschweig / Wiesbaden: Vieweg, 1998

Bark 1996:

CORBA und COM/OLE: Studie über die beiden Standards für objektorientierte Entwicklungen / Michael Bark. – Internet:
http://www.bkm-dv.de/Technische_Umsetzung/317.htm

Bartscher 2001:

Methoden des integrierten Workflow-Managements (IWFM) / Thomas Bartscher. – Paderborn: HNI, 2001

Becker / Kloock / Schmidt / Wäscher 1998:

Unternehmen im Wandel und Umbruch : Transformation, Evolution und Neugestaltung privater und öffentlicher Institutionen / Manfred Becker / Josef Kloock / Reinhart Schmidt / Gerhard Wäscher. – Stuttgart: Schäffer-Poeschel, 1998

Bishop 1998:

Java Gently: Programming Principles Explained / Judy M. Bishop. – second Edition Harlow, England [u.a.]: Addison-Wesley, 1998.

Bühner 1989:

Betriebswirtschaftliche Organisationslehre / Rolf Bühner. – 4. Korr. Aufl. – München, Wien: Oldenbourg, 1989

Daldrup 1995:

Menschengerechte Softwaregestaltung : Konzepte und Werkzeuge auf dem Weg in die Praxis / Uli Daldrup. – Stuttgart: Teubner, 1995

Dix et al. 1993 :

Mensch, Maschine, Methodik / Alan Dix / Janet Finlay / Gregory Abowd / Russell Beale. – New York, London, Toronto, Sydney, Tokio, Singapur, München: Prentice hall, 1993

Freese 1992:

Organisationstheorie: Historische Entwicklung, Ansätze, Perspektiven / Erich Freese. – 2. überarbeite und erweiterte Auflage. – Wiesbaden: Gabler, 1992

Gabler 1998:

Gabler Wirtschaftsinformatik-Lexikon / Eberhard Stickel / Hans-Dieter Groffmann / Karl-Heinz Rau. – 1.Band (A-K), Wiesbaden: Gabler, 1998

Gillner 1984:

Datenbanken auf Arbeitsplatzrechnern / Reinhard Gillner. – München, Wien: Hanser, 1984

Grand / Knudsen 1997:

Java Fundamental Classes Reference / Mark Grand / Jonathan Knudsen. – Sebastopol: O'Reilly & Associates, Inc., 1997

Greenberg 1991:

Computer Supported Cooperative Work and Groupware / Saul Greenberg. – Calgary: EACE Publications, 1991, pp. 1-7

Haberstock 2000:

Executive Information Systems und Groupware im Controlling: Integration durch das prozessorientierte Team-Controlling System (ProTeCos) / Philipp Haberstock. Geleitwort von Ludwig Nastansky. – Wiesbaden: DUV, Deutscher Universitätsverlag, 2000

Halter 1997:

Halter Urs: Workflow Integration im Kreditbereich; in: Österle, Hubert / Vogler, Petra: Praxis des Workflow-Managements – Grundlagen, Vorgehen, Beispiele. – Braunschweig / Wiesbaden: Vieweg-Verlag, 1997

Hofmann / Jobst / Schabenberger 2001:

Programmieren mit COM und CORBA: Einführung in die Architekturen für verteilte Anwendungen / Johann Hofmann / Fritz Jobst / Roland Schabenberger. – München, Wien: Carl Hanser Verlag, 2001

Huth 1998:

Plattform- und werkzeugübergreifende verteilte Organisationsmodellierung für Workflow Management – Entwicklung einer Werkzeugunterstützung zur Modellierung von Organisationssubsystemen für die Arbeit verteilter Teams / Carsten Huth. – Paderborn, 1998. Siehe auch Internet: http://gcc.upb.de

Huth / Nastansky 2000a:

GroupProcess: Partizipatives, verteiltes Design und simultane Ausführung von Ad hoc Geschäftsprozessen, in: Herrad Schmidt (Ed.): Modellierung betrieblicher Informationssysteme, Proceedings der MobIS-Fachtagung 2000, 11. und 12. Oktober 2000, Rundbrief der GI-Fachgruppe 5.10, 7. Jahrgang, Heft 1 / Carsten Huth / Ludwig Nastansky. – Siegen, Universität Siegen, Oktober 2000

Huth / Nastansky 2000b:

GroupProcess: Partizipatives, verteiltes Design und simulative Ausführung von Ad hoc Workflows: Eine Erweiterung eines groupwarebasierten Workflow Management Systems / Carsten Huth / Ludwig Nastansky. – Paderborn, 2000 (siehe http://pdai.inf.tu-dresden.de/Geneme00/folien/Huth.pdf)

Java Spektrum 1999:

Java Spektrum 2/1999. – New York: SIGS Publications, 1999

Kling 1996:

Computerization and controversy: value conflicts and social choices / Rob Kling. – San Diego: Academic Press, Inc, 1996

Knittel 1995:

Technikgestützte Kommunikation und Kooperation im Büro: Entwicklungshindernisse – Einsatzstrategien – Gestaltungskonzepte / Friedrich Knittel. – Wiesbaden: Gabler, 1995

Krüger 1994:

Organisation der Unternehmung / Wilfried Krüger. – 3. verbesserte Auflage. – Stuttgart, Berlin, Köln : Kohlhammer, 1994

Köhler-Frost 1998:

Electronic Office-Systeme: Workflow und Groupwareanwendungen in der Praxis / Wilfried Köhler-Frost. – Berlin: Erich Schmidt, 1998

Maaß et al. 1993:

Software-Ergonomie – Ausbildung in Informatikstudiengängen Bundesdeutscher Universitäten / S. Maaß / D. Ackermann / W. Dzida / P. Gorny / H. Oberquelle / K.-H. Rödiger / W. Rupietta / N. Streitz. – Informatikspektrum 16, 1993, 25-30

Nastansky 1993:

Workgroup Computing : Computergestützte Teamarbeit (CSCW) in der Praxis; neue Entwicklungen und Trends / Ludwig Nastansky. – Hamburg: S + W Steuer und Wirtschaftsverlag, 1993

Nastansky et al. 2000:

Bausteine der Wirtschaftsinformatik: Grundlagen, Anwendungen, PC-Praxis / Joachim Fischer / Werner Herold / Wilhelm Dangelmaier / Ludwig Nastansky / Leena Suhl. – Berlin: S + W Steuer und Wirtschaftsverlag, 2. Auflage, 2000

Nastansky / Hilpert 1994:

The GroupFlow System: A Scalable Approach to Workflow Management between Cooperation and Automation, in: Wolfinger, Bernd (Ed.): Innovationen bei Rechen- und Kommunikationssystemen – Eine Herausforderung an die Informatik, Proceedings of 24th Annual Conference of the German Computer Society during 13th World Computer Congress, IFIP '94 / Nastansky, Ludwig; Hilpert, Wolfgang. – Berlin / Heidelberg etc.: Springer Verlag , 1994, pp. 473 - 479.

OMG:

Was ist CORBA: Author: OMG, Übersetzung: Corba.ch. – Internet: http://www.corba.ch/WasIstCorba.html

OMG 2000:

CORBA Basics / OMG (Object Management Group). – Internet: http://www.omg.org/gettingstarted/corbafaq.htm#WhatIsIt

Partl :

Java Einführung / Hubert Partl. – Internet :www.boku.ac.at/javaeinf/jein1.html

Picot/Reichwald/Wigand 1998:

Die grenzenlose Unternehmung : Information, Organisation und Management; Lehrbuch zur Unternehmensführung im Informationszeitalter / Arnold Picot / Ralf Reichwald / Rolf T. Wigand. – 3. überarbeitete Auflage – Wiesbaden: Gabler, 1998

Ringlstetter 1997:

Organisation von Unternehmen und Unternehmensverbindungen: Einführung in die Gestaltung der Organisationsstruktur / Max. J. Ringlstetter. – München / Wien: Oldenbourg, 1997

Rosenberger 1998:

CORBA in 14 days : /Jeremy Rosenberger. – Indianapolis: Sams Publishing, 1998

Schmidt 1997:

Programmieren im Großen: Modularisierung, Verteilung, Entwurfsmuster – Java Remote Method Invocation / Wolfgang Schmidt. – Internet: http://www.stud.fernuni-hagen.de/q3283267/v_rmi/text/v_rmi.html, 1997

Schulte-Zurhausen 1995:

Organisation / Manfred Schulte-Zurhausen. – München : Vahlen, 1995

Steinbuch 1977:

Kompendium der praktischen Betriebswirtschaft: Organisation / Pitter A. Steinbuch. – Ludwigshafen: Kiehl Verlag , 1977

Steinmetz 1993:

Multimedia-Technologie: Einführung und Grundlagen / Ralf Steinmetz. – Berlin, Heidelberg, New York, London, Paris, Tokyo, Hong Kong, Barcelona, Budapest: Springer, 1993

Steyer 1998:

Java 1.2: Java-Tutorial, Programmierung von Applikationen, die Schnittstelle zu C und C++, Klassenbibliotheken, ausführliche Referenz / Ralph Steyer. – Haar bei München: Markt und Technik, Buch- und Softwareverlag, 1998

Teufel 1995:

Computerunterstützung für die Gruppenarbeit: / Stephanie Teufel / Christian Sauter / Thomas Mühlherr / Kurt Bauknecht. – Bonn: Addison Wesley, 1995

Thielemann 1999:

Integrierte Methodik zur Gestaltung von Leistungserstellungsprozessen mittels Workflowmanagement / Frank Thielemann. – Paderborn: HNI-Verlagsschriftenreihe, Band 58, 1999

Weitzel / Franke / Hittmeyer:

First XML / Tim Weitzel / Jochen Franke / Claus Hittmeyer. – Internet: http://xml.cnec.org/basics/first-xml_index.php

Zeidler / Zellner 1994:

Software-Ergonomie: Techniken der Dialoggestaltung / Alfred Zeidler / Rudolf Zellner. – 2. verb. Auflage, München: Oldenbourg, 1994

Ziegler 1993:

Benutzergerechte Software-Gestaltung: Standards, Methoden und Werkzeuge / Jürgen Ziegler. – München, Wien: Oldenbourg, 1993

Anhang

Inhalt :

A.1 Anwendungsbeispiele des GroupProcess-Projektes 102
A.2 Anwendungsbeispiele der erstellten Werkzeuge 103
A.3 Systemvorraussetzungen und verwendete Hilfsmittel 104
A.3.1 Mindestanforderungen 104
A.3.2 Verwendete Hilfsmittel 105
A.4 Quelldateien im Überblick 106
A.4.1 Quelldateien des OrganizationchartViewers 106
A.4.2 Quelldateien des WorkgroupViewers 107
A.4.3 Quelldateien des Kommunikationspaketes readNotesData 107
A.4.4 Codebeispiel 108

A.1 Anwendungsbeispiele des GroupProcess-Projektes

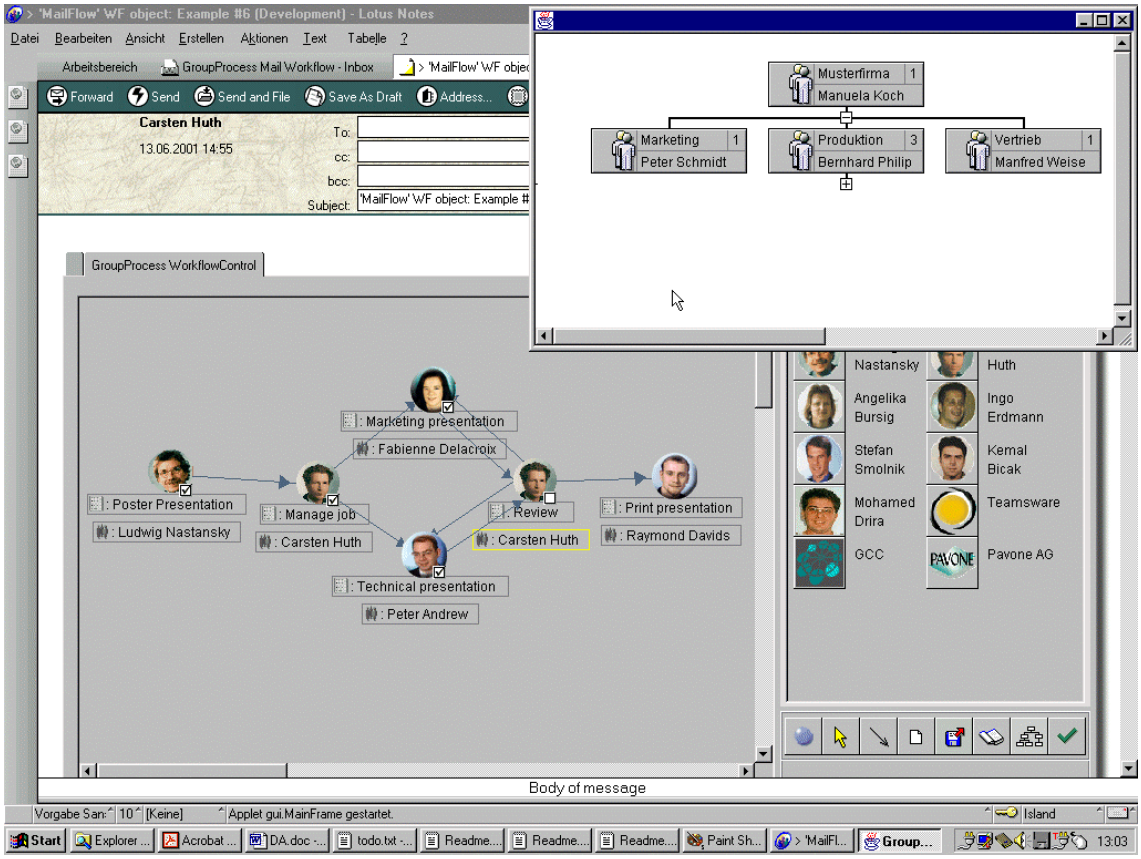

Abbildung 37: Der Ad-hoc-WorkflowModeler in Kombination mit dem OrganizationchartViewer eingebettet in ein Lotus Notes-Dokument

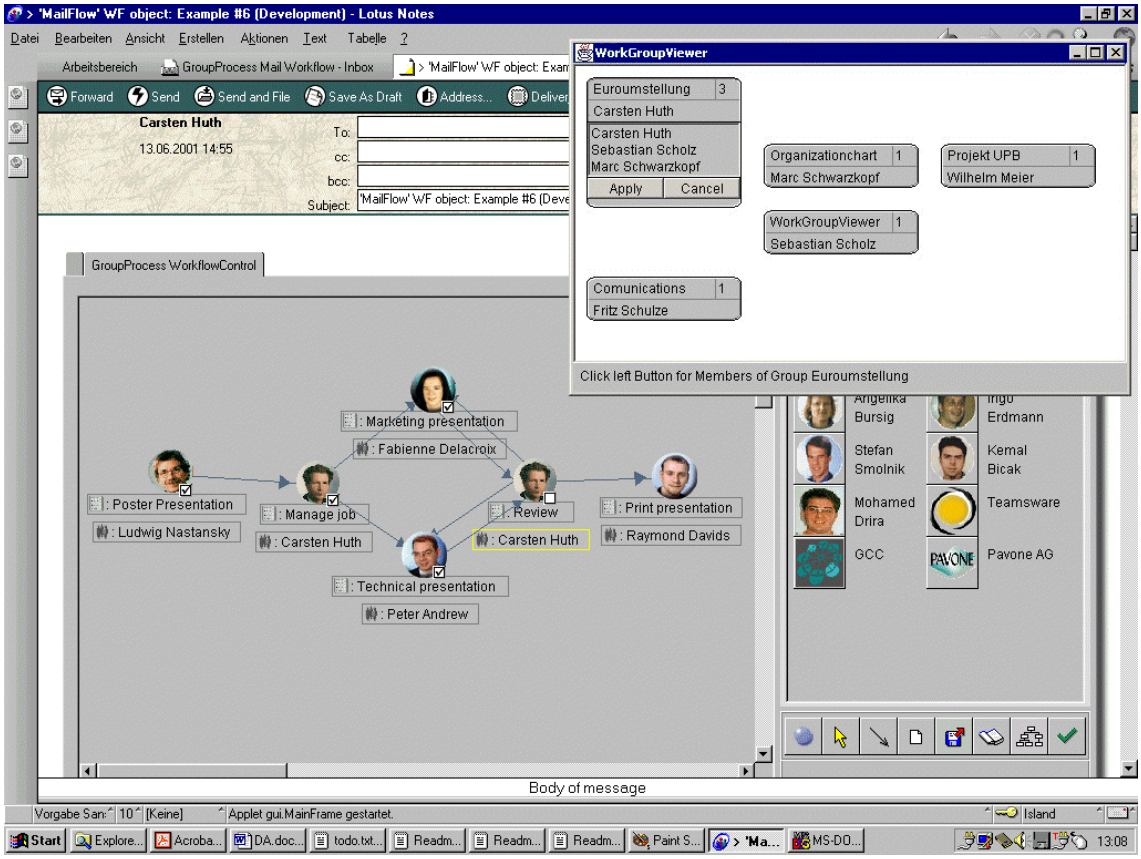

Abbildung 38: Der Ad-hoc-WorkflowModeler in Kombination mit dem WorkgroupViewer eingebettet in ein Lotus Notes-Dokument

A.2 Anwendungsbeispiele der erstellten Werkzeuge

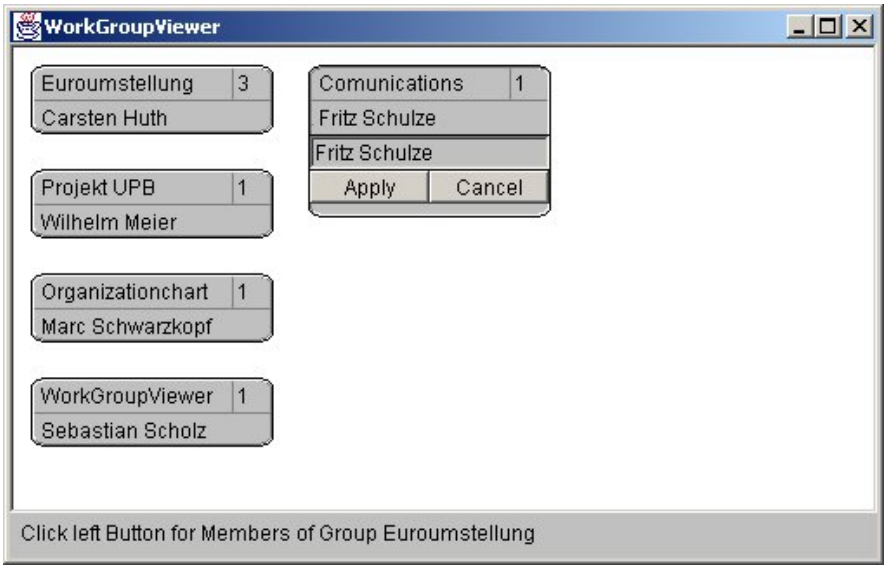

Abbildung 39: Der WorkgroupViewer

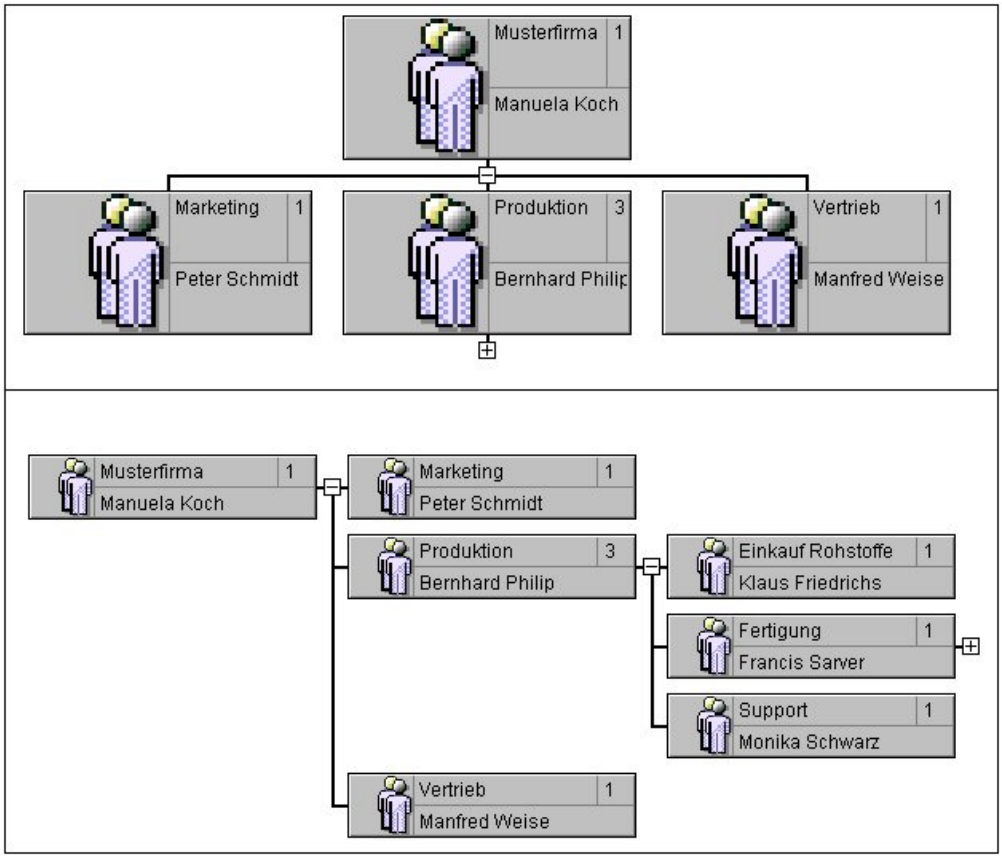

Abbildung 40: Der OrganizationchartViewer in verschiedenen Darstellungen

A.3 Systemvorraussetzungen und verwendete Hilfsmittel

A.3.1 Mindestanforderungen

Um die in Kapitel 5 vorgestellten Werkzeuge zum Einsatz zu bringen, ist es notwendig, die von Lotus Notes bereitgestellten Klassen zur Kommunikation bereitzustellen. Diese Java-Klassen existieren seit der Lotus Notes-Version 5.0.

Zur Ausführung des OrganizationchartViewers und des WorkgroupViewers ist mindestens die Java-Version 1.1.8 notwendig. Diese ist in Lotus Notes-Version 5.0 bereits integriert. Bei der Nutzung der Werkzeuge über das Internet mit einem Browser in Kombination mit einem Lotus Domino-Server ist es empfehlenswert, den Microsoft Internet Explorer ab der Version 5.0 oder den Netscape Navigator ab der Version 4.6 einzusetzen. Zusätzlich ist es natürlich möglich, den in Lotus Notes integrierten Browser zu benutzen.

A.3.2 Verwendete Hilfsmittel

Eingesetzte Entwicklungswerkzeuge:

- Lotus Development Corporation
  Lotus Notes-Version 5.0.6a
- Sun Microsystems
  JDK 1.1.8
  JDK 1.3
- Webgain Inc.
  Visual Café 4.1a
- Inprise Corporation
  Jbuilder 5.0
- TogetherSoft Corporation
  Together ControlCenter 5.02

Weitere genutzte Softwareprodukte:

- PAVONE AG
  PAVONE Enterprise Office Release 5.01
  PAVONE OrganizationModeler Release 5.01
  PAVONE ProcessModeler Release 5.01

A.4 Quelldateien im Überblick

A.4.1 Quelldateien des OrganizationchartViewers

Pfad/Dateiname	Größe (Byte)	Funktion
OrganizationchartViewer.java	13.124	Verwaltet das GUI
DataTransfer.java	197	Kommunikationsinterface zum Ad-hoc-WorkflowModeler
ThreadNotifier.java	158	Interface zur Überwachung von Mausklicks
ADT/Tuple.java	183	Datenhaltungsklasse
ComputeRelations/computeRelations.java	6.237	Berechnung der Abteilungs-struktur als Baum.
Config/ConfigFrame.java	2.285	Klasse zum Halten der Konfigurationsdaten
Threads/WaitonClick.java	566	Thread zur Überwachung von Mausklicks
Unit/PersonsD.java	2.681	Panel zur Anzeige von Personen einer Abteilung
Unit/GUIUnit.java	5.381	Hält graphische Daten einer Abteilung
UnitGraph/HierarchialUnit.java	11.095	Hält die Hierarchiestruktur
Gesamtgröße des Codes	41.907	

Tabelle 3: Klassenstruktur des OrganizationchartViewers

A.4.2 Quelldateien des WorkgroupViewers

Pfad/Dateiname	Größe (Byte)	Funktion
WorkgroupViewer.java	4.426	Verwaltet das GUI
HelpArea.java	423	Panel zur Anzeige der Online-Hilfe
PersonsPanel.java	3.484	Panel zur Anzeige der Personen einer Arbeitsgruppe
WorkGroup.java	3.513	Panel zur Anzeige der Arbeitsgruppen
Gesamtgröße des Codes	11.846	

Tabelle 4: Klassenstruktur des WorkgroupViewers

A.4.3 Quelldateien des Kommunikationspaketes readNotesData

Pfad/Dateiname	Größe (Byte)	Funktion
DataHandle.java	2.800	Stellt Datenbankverbindung her.
NamesData.java	3.105	Liest aus der Names Datenbank
Group.java	3.155	Speichert Gruppendaten
Person.java	3.333	Speichert Personendaten
OrganizationData.java	3.764	Liest aus Organisationsdatenbank
Unit.java	4.393	Speichert Abteilungsdaten
Gesamtgröße des Codes	20.550	

Tabelle 5: Klassenstruktur des Kommunikationspackages

A.4.4 Codebeispiel

```
public boolean mouseDown(Event e,int x,int y) {
    int NoOfUnitElems = ComputeRelations.noOfElems(Units);
    int i;
    int curUnitX;
    int curUnitY;
    int curUnitHeight;
    int curUnitWidth;
    int SwitchXO;
    int SwitchYO;
    boolean UnitHit;
    HierarchialUnit   curUnit=null;

    if (CurWatch != null) {
        remove (CurWatch);
        if (!CurWatch.isVisible()) {
          Watchisactive=false;
          System.out.println("Not active");
        }
    }

    if (ThreadStarted) {
          w.stop();
          ThreadStarted=false;
        }

    if (e.clickCount==2 && SelectedUnit.getVisible()) {
        //hier wird überprüft, ob beim 2. Klick die gleiche Unit getroffen wurde
        for(i=0;i<NoOfUnitElems;i++) {
            curUnit = (HierarchialUnit) Units.elementAt(i);
            curUnitHeight = curUnit.getHeight();
            curUnitWidth  = curUnit.getWidth();
            curUnitX        = curUnit.getUpperLeftX();
            curUnitY        = curUnit.getUpperLeftY();

            if(x >= curUnitX && x <= (curUnitX+curUnitWidth) &&
                y >= curUnitY && y <= (curUnitY+curUnitHeight)) {
                break;
            }
        }
        .
        .
```

Abbildung 41: Codebeispiel der mouseDown-Routine im OrganizationchartViewer

Der hier gezeigte Code ist ein Ausschnitt aus der mouseDown-Routine der Klasse OrganizationchartViewer.java, welche für den Aufbau und den Ablauf des GUI zuständig ist. Im Detail wird hier überprüft, ob eine Abteilung einzeln oder doppelt mit der Maus angeklickt wurde. Die Abfrage wird benötigt, um die entsprechende Aktion einzuleiten, die entweder die Personenübersicht öffnet oder die Abteilungsdaten an den Ad-hoc-WorkflowModeler übergibt.

Eidesstattliche Erklärung des Marc Schwarzkopf

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Minden, den 27.10.2001

Eidesstattliche Erklärung des Sebastian Scholz

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Hofgeismar, den 27.10.2001